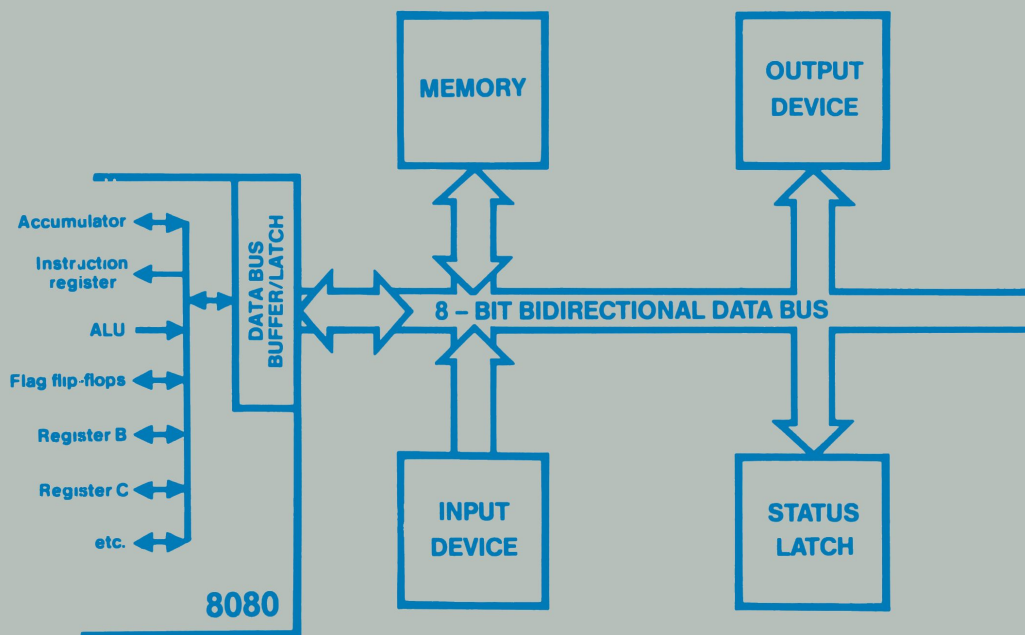
# il BUGBOOK® VI

ESPERIMENTI INTRODUTTIVI ALL'ELETTRONICA DIGITALE, ALLA PROGRAMMAZIONE E ALL'INTERFACCIAMENTO DEL MICROCOMPUTER 8080A

EDIZIONE ITALIANA

JACKSON ITALIANA EDITRICE

# il BUGBOOK VI

ESPERIMENTI INTRODUTTIVI ALL'ELETTRONICA
DIGITALE, ALLA PROGRAMMAZIONE
E ALL'INTERFACCIAMENTO DEL MICROCOMPUTER 8080A


**PETER R. RONY**
*Department of Chemical Engineering*

**DAVID G. LARSEN**
*Department of Chemistry*

Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061

e

**JONATHAN A. TITUS**
*Tychon, Inc.*
Blacksburg, Virginia 24060


Versione italiana


**ALDO CAVALCOLI**
**VALERIO SCIBILIA**
*Mipro s.r.l.*
Via Carducci, 15
20123 Milano


JACKSON
ITALIANA
EDITRICE

Piazzale Massari, 22
20125 Milano

# INTRODUZIONE ALL'EDIZIONE ITALIANA

La serie dei Bugbook ha rappresentato negli Stati Uniti una pietra miliare nella divulgazione e nell'insegnamento dell'elettronica digitale e delle tecniche di utilizzo dei microprocessori.

I testi sono sorti da esperienze didattiche degli autori, che svolgono attività di ricercatori e docenti presso il Virginia Polytechnic Institute and State University.

Questa esperienza scolastica ad alto livello è una delle caratteristiche più interessanti della serie dei Bugbook, che possono anche essere utilizzati come validi manuali di autoistruzione.

Nella preparazione dell'edizione italiana, abbiamo cercato il più possibile di mantenere lo stile colloquiale degli autori, che del resto dal punto di vista umano sono molto estroversi ed entusiasti della loro attività di ricercatori.

Abbiamo mantenuto molti termini tecnici inglesi sia per il diffuso utilizzo che se ne fa nella corrente terminologia elettronica, sia per familiarizzare il lettore con un certo linguaggio, che sempre più è presente nella letteratura tecnica.

Le apparecchiature descritte nei volumi e utilizzate negli esperimenti hanno avuto origine dall'attività di ricerca degli autori, secondo precise esigenze didattiche; queste apparecchiature sono prodotte dalla E-L Instruments.

Un particolare ringraziamento va a quanti hanno collaborato alla realizzazione della versione italiana dei Bugbook, l'Ing. Ettore Valsecchi, la Sig.na Daniela Fornari, la Sig. ra Rosaria Lucano, il Sig. Marcello Longhini, la Sig.ra Francesca Di Fiore, ed i Sig.ri Giampietro Zanga e Paolo Reina ideatori della versione italiana dei Bugbook.

Un ringraziamento va anche alle organizzazioni che hanno reso possibile e agevolato il nostro lavoro:

MICROLEM S.a.S. - Via C. Monteverdi, 5 - 20131 Milano - distributore per l'Italia dell'apparecchiatura della E-L Instruments.

MIPRO S.r.l. - Via Carducci, 15 - 20123 Milano - società di consulenza minicomputers-microprocessors.

ELETTRONICA OGGI - Piazzale Massari, 22 - 20125 Milano - rivista mensile di elettronica, microelettronica e automazione professionale.

*Aldo Cavalcoli*
*Valerio Scibilia*

# SOMMARIO

**CAPITOLO 18 - IL SET DI ISTRUZIONI 8080A**



**CAPITOLO 19 - LE TECNICHE DI BUS DATI NELL'USO DI DISPOSITIVI THREE-STATE**

## CAPITOLO 20 - INTRODUZIONE ALLE TECNICHE DI I/O TRAMITE L'ACCUMULATORE



## CAPITOLO 21 - INTRODUZIONE ALLE TECNICHE DI INGRESSO/USCITA MEMORY-MAPPED

## CAPITOLO 22 - L'INGRESSO/USCITA DEL MICROCOMPUTER: ALCUNI ESEMPI



## CAPITOLO 23 - FLAG E INTERRUZIONI

**APPENDICI**

# PREFAZIONE

Benvenuti nella nuova rivoluzione elettronica. In 10 anni la tecnologia dei circuiti elettronici ha trasformato i circuiti integrati digitali da costosi componenti per semplici funzioni logiche, a componenti ad alta complessività, contenenti fino a 10.000 transistor. Da questa rivoluzione è nato il computer-su-singolo-chip! Questo nuovo componente contiene tutto quello che è richiesto da un computer digitale (unità centrale, memoria a lettura/scrittura, memoria ROM, circuiteria di interfaccia). Fra pochi anni sarete in grado di acquistare una manciata di questi componenti a prezzi impensabili. Si ritiene che intorno al 1980 esisterà circa un milardo di microcomputer in circolazione: per avere un'idea del significato esplosivo di questa cifra, basta pensare che attualmente in America vi sono solo 250.000 tra minicomputer e grandi sistemi di elaborazione dati. Una rivoluzione nel campo dei computer? Sicuramente.

Nell'istruzione poi, pensiamo che la nuova rivoluzione elettronica creerà importanti cambiamenti ed opportunità:

- Un numero sempre maggiore di studenti dovrà imparare l'elettronica digitale ed in particolare l'utilizzo dei microcomputer; studenti che diventeranno ingegneri, chimici, biologi, fisici, tecnici agrari, biochimici, psicologi sperimentali.

- I corsi teorici sull'algebra Booleana, sulle mappe di Karnaugh, ed altri argomenti equivalenti, perderanno di importanza per tutti quegli studenti che sono interessati alla tecnologia digitale.

- Gli studenti che si occupano di Scienza dei Calcolatori si troveranno sempre più nella necessità di apprendere nozioni di elettronica digitale: molti studenti avranno il loro proprio microcomputer.

- Centinaia di microcomputer saranno disponibili nell'ambito delle Università; forse migliaia.

- I corsi di telecomunicazioni e controlli digitali cresceranno in importanza.

A fronte di questi cambiamenti, una cosa resterà essenzialmente invariata: il tempo che gli studenti passano a scuola. Gli insegnanti dovranno affrontare il problema di armonizzare gli argomenti sopra indicati con i corsi esistenti, per un inevitabile aggiornamento; sarà anche necessaria la creazione di nuovi corsi e la fusione di altri già in programma, per fornire contenuti didattici validi per l'evoluzione tecnologica ed industriale in atto.

I BUGBOOKS costituiscono una serie di volumi sull'elettronica digitale, sull'interfacciamento dei microcomputer e sulla programmazione dei microcomputer, si tratta di un tentativo di integrazione dei tre argomenti fondamentali succitati in un corso unificato. Questi volumi sono orientati verso esperimenti di laboratorio, che consideriamo la via migliore per esprimere l'eccezionalità e l'importanza della nuova rivoluzione

elettronica. Ai tre argomenti indicati viene data la stessa importanza: imparerete come si programma un microcomputer, come lo si interfaccia verso dispositivi esterni e come i dispositivi esterni operano da un punto di vista digitale. Saranno illustrati importanti concetti di elettronica digitale sia da un punto di vista circuitale, collegando opportuni circuiti integrati, sia da un punto di vista software, realizzando programmi per microcomputer.

Per il lettore di questi volumi, non è necessaria una particolare precedente esperienza in elettronica digitale. Come prima cosa saranno trattati i microcomputer ed i circuiti integrati come *moduli funzionali*. Da questa esposizione verranno apprese in modo graduale le caratteristiche operative di base. Non discuteremo come questi dispositivi sono stati costruiti, in quanto la tecnologia è alquanto sofisticata e si modifica rapidamente.

I Bugbooks sono testi "laboratory oriented", nell'ambito di una serie di libri caratterizzati da un approccio all'elettronica digitale a nostro avviso differente. Piuttosto che iniziare, come è abitudine, con esperimenti sui *componenti elettronici*, come *resistori, condensatori, diodi, transistor,* con questi volumi vi introduciamo direttamente ai concetti relativi ai *circuiti integrati,* i cosiddetti "bugs" dei nostri Bugbooks. Vi forniamo poi immediatamente i concetti di: *switch logici, indicatori a LED, generatori di impulsi, display;* vi mostriamo come utilizzare queste funzioni ausiliarie e vi sottoponiamo esperimenti relativi ai collegamenti tra i circuiti integrati e questi dispositivi.

Una volta in possesso dei concetti base dell'elettronica digitale e delle modalità di collegamento dei vari circuiti, vi introduciamo all'utilizzo di chip e sistemi digitali più complessi. Imparerete come utilizzare un ricevitore/trasmettitore universale asincrono (UART), come interfacciarsi ad un microcomputer basato sul microprocessor 8080, unitamente ai concetti base della programmazione e dell'interfacciamento di un microcomputer. Come lavorare con un UART è trattato nel Bugbook II, mentre le tecniche di interfacciamento dell'8080 sono oggetto dei Bogbooks V e VI.

I Bugbooks V e VI sono orientati ad un insegnamento sperimentale sull'elettronica digitale. Come già detto, il nostro scopo è integrare l'elettronica digitale, l'interfacciamento dei microcomputer, la programmazione dei microcomputer in un singolo corso unificato. I concetti relativi alle tecniche di programmazione ed interfacciamento sono discussi unitamente ai principi di elettronica digitale, verificati sperimentalmente tramite l'utilizzo dei più noti chip, quali il 7400, 7402, 7404, 7442, 7475, 7490, 7493, 74121, 74125, 74126, 74150, 74154, 74181, 74193.

Noi pensiamo che l'elettronica digitale tende sempre più verso l'utilizzo dei microcomputer. Di conseguenza vi sarà un considerevole sforzo in campo didattico per introdurre l'utilizzo dei microcomputer, come del resto sta già accadendo in molte Università ed Istituti Tecnici. Quanto detto va oltre l'ambiente scolastico per interessare professionisti e tecnici desiderosi di aggiornarsi nell'elettronica digitale. I Bugbooks V e VI sono diretti anche a loro.

I Bugbooks V e VI sono da considerare testi per autodidattica. Nei vari capitoli vi sono le risposte a tutte le domande poste e riepiloghi finali per dei concetti trattati. Chi può utilizzare con profitto questi libri? Riteniamo non sia necessario una particolare conoscenza in elettronica. Se siete in grado di organizzare e comprendere i nuovi concetti, di estrapolare "know-how" da nuove situazioni, di realizzare gli esperimenti suggeriti, trarrete il massimo vantaggio dai Bugbooks. I Bugbooks V e VI si configurano molto bene come programma di autodidattica per professionisti che desiderano aggiornarsi nel campo della elettronica digitale.

La nostra esperienza nell'utilizzare i Bugbooks durante i corsi universitari è stata entusiasmante. Stralci dei Bugbooks sono stati tradotti in tedesco, giapponese, francese, italiano, malese. L'attuale traduzione completa in italiano è la prima eseguita in Europa.

Abbiamo poi osservato la carenza di mezzi educazionali in molti campi specifici dell'elettronica digitale; a questo proposito abbiamo definito dei nuovi volumi della serie, quali il "The 555 Timer Applications and Sourcebook, with Experiments" scritto da Howard Berlin, nella versione italiana sempre dalla Jackson Editrice. Altri volumi della serie Bugbooks, sempre pubblicati dalla Jackson in versione italiana, sono: "The Design of Active Filters, with Experiments" e "Designing with Operational Amplifiers, with Experiments".

Brevi corsi sull'interfacciamento dei microcomputer sono organizzati dalla MICROLEM in collaborazione con la CPM ed il VIRGINIA POLYTECHNIC INSTITUTE and State University. Questi corsi, che si sviluppano su 5 giorni, comprendono una giornata di revisione di elettronica digitale, tre giornate e mezza sull'interfacciamento e i principi di programmazione dei microcomputer, e una giornata dedicata alle applicazioni dei microcomputer. I partecipanti utilizzano, per tutta la durata del corso, il microcomputer MMD-1, per testare sia i programmi che i circuiti di interfaccia presentati nei Bugbook V e VI.
Per ulteriori informazioni, rivolgersi alla segreteria del "Blacksburg Continuing Education Courses in Italy" tel. (02) 2710465.

Vogliamo ringraziare chi continua ad appoggiare il nostro sforzo educativo. Mr. Murray Gallant e la E-L Instruments che hanno finanziato lo sviluppo del microcomputer MMD-1 di Jon Titus; Mr. Veltri che ha realizzato delle eccellenti fotografie del nostro materiale.

*Peter R. Rony, David G. Larsen, Jonathan A. Titus*
Blacksburg, Virginia 24060

# CAPITOLO 16

# COSA VUOL DIRE INTERFACCIARE?

## INTRODUZIONE

Questo capitolo fornirà alcuni principi relativi all'interfacciamento e vi darà le definizioni di alcuni concetti fondamentali.

## OBIETTIVI

Alla fine di questo capitolo sarete in grado di:

- Sapere le differenze tra un microcomputer e un microprocessor
- Dare la definizione di processore di dati.
- Sapere la differenza tra software e hardware
- Dare la definizione di controller
- Discutere tutta la gamma dei dispositivi relativi ai computer dalla logica cablata ai grandi sistemi di elaborazione dati
- Descrivere i sistemi di bus dei microcomputer basati sull'8080A
- Elencare cinque importanti linee di segnali di controllo del microcomputer MMD-1
- Dare la definizione di sincrono, di dispositivo di I/O, di CPU e di memoria

**LA RIVOLUZIONE DELLA MACCHINA INTELLIGENTE**

Nel Bugbook V vi abbiamo dato delle informazioni sull'elettronica digitale di base di cui avrete bisogno per *interfacciare* un microcomputer basato sull'8080A. Avremmo avuto altri soggetti da trattare - Three state bussing, registri di shift, unità aritmetico-logiche e buffer - ma abbiamo preferito parlarvi prima delle operazioni logiche, AND, OR, NAND, NOR e OR-Esclusivo; delle caratteristiche di gating delle quattro porte base; dei decodificatori; dei latch e dei flip-flop; contatori; multivibratori monostabili; dispositivi di ingresso-uscita come switch, generatori di impulsi, clock e display; codici digitali; dei termini importanti come, strobe, enable disable, gate e inhibit.
Prima di passare all'argomento interfacciamento, crediamo sia meglio dirvi per quali motivi avrete bisogno di interfacciare il microcomputer.

Per coloro che possono reperire la pubblicazione *Business Week* della Mc Graw-Hill, diciamo subito di leggere, sul numero del 5 luglio 1976, l'articolo "The smart Machine revolution: Providing products with brainpower"; per gli altri riportiamo alcuni passi di questo articolo:

- _È la seconda rivoluzione industriale', dice Sidney Webb, vice-presidente esecutivo della TWC Inc. 'Si moltiplica la capacità del cervello umano con la stessa proporzione con cui la prima rivoluzione industriale moltiplicò la forza muscolare!'

- Il propulsore della rivoluzione è il microprocessor, detto anche computer in un singolo chip, una piccola piastrina di silicio che è il nucleo artimetico e logico di un computer. La prima ondata di prodotti con un cervello costituito da un microprocessor sta cominciando ora a invadere il mercato, e lo dimostra il fatto che mai prima d'ora vi erano strumenti così potenti per costruire 'smart machines' - macchine che oltre alle loro normali funzioni possono prendere decisioni, calcolare e memorizzare. Questa ondata di macchine intelligenti comprende:

  Orologi intelligenti
  Bilance intelligenti
  Telefoni portatili intelligenti
  Sistemi di inscatolamento intelligenti
  Giochi televisivi intelligenti

  Una marea di questi prodotti intelligenti si sta riversando sul mercato, modificandolo drasticamente per quanto riguarda i prodotti di consumo, industriali e commerciali. Il computer in un singolo chip, che fornisce il cervello a queste macchine intelligenti, fa nascere nuove aziende e centinaia di nuove industrie, mentre sarà causa di morte per altre".

- "La spiegazione di questa ondata di vendite di microprocessori e dell'innovazione di queste macchine intelligenti, è tutta nel prezzo."
  C. Lester Hogan, vice presidente della Fairchild Camera of Instruments Corp. ha posto in risalto questo aspetto alla Convention di Boston, quando ha estratto di tasca 18 microprocessori e li ha gettati in mezzo al pubblico dicendo: 'Questo è l'equivalente di 18 milioni di dollari di computer, o almeno lo era 20 anni fa'. Ha spiegato che i suoi microprocessori da 20 dollari hanno la stessa potenza del primo computer commerciale della International Bussiness Machines che, negli anni '50, costava 1 milione di dollari. 'Quello che volevo sottolineare - ha detto - è che oggi i computer sono praticamente gratuiti!'

  Fino ad un anno fa, questi microprocessori da 20 dollari ne costavano 100,

e questo improvviso abbassamento dei prezzi ha permesso ai progettisti di dare il via al diluvio di prodotti intelligenti. Il passaggio dall'elettronica convenzionale, quella dei circuiti integrati, alla MPU, ha permesso di ridurre i tempi di progettazione e i costi di fabbricazione perché da sola, la MPU, sostituisce centinaia di circuiti integrati e altro. Una volta che la MPU è progettata per un particolare prodotto, garantisce sul mercato degli enormi vantaggi; le funzioni che un prodotto svolge possono essere cambiate non riprogettando, con alta spesa, la parte elettronica, ma modificando solo le istruzioni, cioè il software, che sono memorizzate nella memoria della MPU. Si possono aggiungere altre funzioni con una piccola spesa supplementare e inoltre nuove macchine intelligenti possono svolgere lavori che prima non erano meccanizzabili per fattori economici."

- "Dei nuovi prodotti guidati da un computer su un solo chip, i più sorprendenti sono quelli destinati al largo consumo. I microprocessor entrano in ogni casa, automobile, apparecchio o prodotto per il consumatore così come entrano in altri prodotti. 'Nel 1980 ci saranno in ogni casa tra 7 e 10 microprocessori'; è la previsione fatta da Andrew A. Perlowski, che dirige le ricerche sui microprocessori alla Honeywell Inc. La sua società sta lavorando sul trattamento dell'energia e sui sistemi di sicurezza per la casa."

- "Nell'industria, il computer in un singolo chip, sta abbassando i costi dell'intelligenza elettronica tanto da diventare l'elemento meno costoso delle macchine intelligenti, e si sta avvicinando sempre di più il giorno in cui ci sarà la fabbrica completamente automatizzata costruita collegando tra loro macchine intelligenti, sensori e tutti gli altri strumenti che servono per la raccolta dati e i sistemi di controllo.

  L'automazione delle fabbriche si è evoluta molto lentamente anche perché gli imprenditori non volevano che un intero impianto andasse fuori uso solo perché un bit era errato. 'Il vantaggio MPU è che decentra il controllo in tante piccole parti in modo che, se una di queste si rovina, non si blocca tutto l'insieme,' dice Sheldon G. Lloyd, vice presidente della progettazione alla Fisher Controls Co. 'Il microprocessor rende economicamente possibile sviluppare e costruire dei sistemi gerarchici".

  In una struttura gerarchica, i microprocessori delle macchine di produzione intelligenti, sono collegati ad un minicomputer supervisore che raccoglie e invia relazioni sulla gestione e informazione sullo stato ad un computer centrale della fabbrica. In cima alla gerarchia si trova il grande computer della società che, una volta collegato ai sistemi delle singole fabbriche, può produrre in un secondo delle relazioni finanziarie sull'andamento dell'intera società.

- "Molti lavori che vengono oggi eseguiti con un microprocessor fino a ieri erano considerati troppo piccoli per essere automatizzati. La Dow Chemical Co. sta prendendo in considerazione l'uso delle MPU per quei lavori dove sono richiesti dei calcoli, ma che non sono tanto complessi per giustificare un minicomputer; dice Charles R. Honea responsabile della strumentazione di processo della divisione texana della Dow. Per esempio la Dow usa dei microprocessori per calcolare il flusso di etilene negli impianti, operazione che, quando veniva fatta normalmente, richiedeva almeno una dozzina di persone. 'E inoltre' dice il sig. Honea, aveva sempre un giorno di ritardo, cioè non si riusciva a sapere quanto etilene si era consumato in quella giornata".

  Le industrie basate su veri e propri processi produttivi hanno un'anima conservatrice in parte dovuta al bisogno di affidabilità nel controllo dei macchinari per permettere agli impianti di funzionare ventiquattro ore su ventiquattro. Di solito occorrono

cinque o sei anni perché una nuova tecnologia diventi di uso comune in questo tipo di industrie. Nicholas P. Scallon, vice-presidente marketing della Fisher Control dice che per i microcomputer non è stato diverso, solo che essi hanno accelerato l'automazione suddividendo i grossi controlli in piccoli segmenti. Invece di tentare di controllare l'intero sistema, usiamo di solito un microprocessor.

Per controllare ogni singolo elemento come un bollitore, un evaporatore o un convertitore catalitico".

- Il software non è solo il problema più grosso per gli utilizzatori della MPU, ma è il problema più costoso. "Oggi il software per un microcomputer costa come quello di un mini' dice Richard Marley, consulente del New Hampshire che ha progettato prodotti intelligenti per molte piccole industrie; egli afferma che le spese per il software sono di circa 100.000 dollari per ogni progetto in cui l'hardware costa circa 20.000 dollari".

- "Probabilmente il microprocessor non è entrato in nessun settore con la penetrazione con cui è entrato nel campo della strumentazione. Il consulente di tecnologie Lynwood D. Eikrem, afferma che negli ultimi due anni, gli strumenti sul mercato che usano microprocessori, sono passati dal 20 al 50%. Le industrie stanno spingendo verso i microprocessori e quelle che non si muoveranno al più presto perderanno il mercato.

  All'inizio lo sforzo più grande era rivolto verso gli strumenti digitali di test, come i voltmetri, i contatori e i sintetizzatori di frequenza, e verso gli strumenti analitici come gli spettrometri e i cromatografi. "Probabilmente già dal 1980 il 9% degli strumenti digitali che costano più di 2000 dollari, useranno dei microprocessori dice Galen W. Wampler".

- "Nel futuro e per molti anni, i prodotti e le macchine intelligenti si diffonderanno ad una velocità sempre maggiore; il software diventerà sempre più accessibile di modo che ognuno potrà programmarsi un microcomputer. Le scuole sforneranno un gran numero di giovani ormai familiarizzati coi microprocessori e quindi in grado di costruire dei prodotti che ne facciano uso. L'industria produttrice di semiconduttori continuerà a sviluppare componenti sempre più potenti.' Nei prossimi 5 o 10 anni saremo in grado di sviluppare oltre 1 milione di dispositivi in un singolo chip', è la previsione che fa Richard L. Petritz, vice presidente della New Business Resources. Questo significa che la potenza sia di un grande computer che di un minicomputer con una grande memoria, saranno disponibili in un singolo chip".

- "Il tempo necessario per sviluppare un prodotto intelligente è così breve e i costi di avviamento sono così bassi che sorgeranno miriadi di nuove industrie che formeranno dei giovani che svilupperanno prodotti basati sulla MPU, dice Hogan della Fairchild, e Petritz dice che 'la MPU ridurrà le applicazioni dell'elettronica allo scrivere programmi per computer e che anche la persona media potrà venire istruita a programmarli'.

  Tutto ciò mette in pericolo le industrie già affermate. Inoltre 'i costruttori dovranno prendere in considerazione i microcomputer o vedranno i loro prodotti diventare obsoleti' ammonisce Donald V. Kleffman, marketing manager della Ampex Corp. Lessler della NCF dice: 'Ci saranno molte nuove industrie che si approprieranno della tecnologia della MPU e rimpiazzeranno alcune delle vecchie società. Molte industrie verranno travolte'.

In uno dei prossimi capitoli parleremo più dettagliatamente di alcuni di questi punti.

## MICROPROCESSOR E MICROCOMPUTER

Abbiamo una certa difficoltà a trovare una buona definizione del termine *computer digitale.* Cercando qualche buona definizione abbiamo trovato un paio di paragrafi scritti da Donald Eadie nel suo libri "Introduction to the Basic Computer" che possono fornire qualche delucidazione sul significato del termine *processore*.

> "Questo capitolo serve come introduzione generale al campo dei dispositivi digitali, con particolare enfasi su quei dispositivi detti *computer* o, molto più propriamente, *elaboratori di dati* (data processor). Il nome elaboratore è molto più inclusivo perché le macchine moderne che ricadono in questa classificazione, non eseguono solo dei calcoli come si intende comunemente, ma compiono altre funzioni con i dati che entrano od escono. Per esempio, gli elaboratori possono raccogliere dati da più sorgenti, ordinarli, adattarli e stamparli. Nessuna di queste funzioni implica delle operazioni aritmetiche quali quelle solitamente associate con i dispositivi di calcolo, eppure si parla, anche in questo caso di computer, cioè calcolatori".
>
> "Quindi, per quanto interessa noi, un computer è in realtà un elaboratore, anche se le operazioni da compiere sui dati possono richiedere operazioni aritmetiche come la somma. Quanto abbiamo detto può forse spiegare perchè un certo numero di imprecisioni sono entrate ormai nel linguaggio comune e perchè esiste una certa confusione tra i termini *computer* ed *elaboratore*. Questi due termini vengono oggi usati con tale frequenza che non è immediato determinare esattamente cosa significano".

Eadie ha definito nel modo seguente il termine *elaboratore*:

*Elaboratore:* dispositivo digitale che esegue una elaborazione sui dati. Può essere un computer, ma in senso lato può raccogliere, distribuire, assimilare, analizzare e organizzare i dati. Queste operazioni non sono quindi necessariamente di calcolo. Elaboratore è un termine più ampio di computer.[15].

Ha inoltre tentato di definire il termine *microprocessor* nel modo seguente:

*Microprocessor:* elaboratore estremamente piccolo

Oggi però non si definisce in questo modo il microprocessor. Quando i costruttori di semiconduttori saranno in grado di costruire un intero computer in un singolo chip, compresa la memoria e le porte di I/O, allora il termine microprocessor avrà il significato che abbiamo appena detto.

Per il momento esiste ancora una distinzione fra *microprocessor* e *microcomputer.* Riportiamo quanto dice al riguardo il "microprocessor Handbook" della Texas Instruments Inc.

> "Questa lezione comincia con la parola "micropocessor". Per alcuni microprocessor è e significa microcomputer, per altri la parola microprocessor e microcomputer, sono diverse. Per costoro, 'microprocessor' è il termine più ampio e generico per descrivere un sistema elettronico molto piccolo capace di eseguire degli specifici programmi. Quindi il microcomputer è un'applicazione del microprocessor[16]".

Al momento noi consideriamo un microprocessor come un singolo circuito integrato che contiene circa il 75% della potenzialità di un piccolo computer. Di solito non può fare nulla senza l'aiuto e il supporto di altri chip e di una memoria.

Al contrario, un microcomputer è un sistema di calcolo completamente operativo, basato su un microprocessor. Questo sistema contiene, oltre al microprocessor, delle memorie, dei latches, dei contatori, dei dispositivi di *ingresso/uscita*, dei buffers e un alimentatore.

Riguardo a questo argomento, vogliamo riportare una parte di un articolo di Laurence Altman apparso su *Electronics* il 18 aprile 1974:

> "Cos'è un microprocessor...ma, prima di tutto vediamo cosa non è. Un microprocessor non è un computer, ma solo una parte di esso. Per fare un computer partendo da un microprocessor bisogna aggiungere una memoria per i programmi di controllo e dei circuiti di *Ingresso/Uscita* per far funzionare le periferiche.
>
> "Che cosa è un microprocessor; allora, è la parte di un piccolo computer che esegue i controlli ed elabora. Più precisamente, è quel tipo di processore che può essere costruito con la circuiteria LSI MOS, di solito in singolo chip. Come tutti i processori di un computer, i microprocessori possono trattare dati logici e aritmetici sotto il controllo di un programma. In ogni caso si distinguono sia dal processore di un minicomputer per l'uso della LOGICA LSI coi suoi bassi costi e consumi, sia dagli altri dispositivi LSI per la possibilità di essere programmati".
>
> "Detto in parole povere, se un minicomputer equivale ad 1 cavallo vapore, il micro processore più la sua circuiteria di supporto equivale a 1/4 di cavallo vapore, ma, siccome la tecnologia LSI si sviluppa sempre più, diventerà certamente più potente. Già oggi si stanno sviluppando processori su singolo chip bipolari e CMOS-on-Sapphire, con potenzialità di un minicomputer".

Come esempio di quale sarà il prossimo futuro, vogliamo riportare le parole di un annuncio fatto nel giugno 1976 da *Digital design*:

> "PROCESSORE, MEMORIA ROM E RAM IN UN SINGOLO CHIP"
>
> La possibilità di avere un chip microprocessor con RAM e ROM comprese, fa avvicinare il giorno in cui i progettisti potranno cambiare il tipo di applicazione modificando solamente i collegamenti di un chip invece di caricare un nuovo programma.
>
> Un microcomputer di questo tipo, il cui costo si aggira sui 10 dollari per quantità di 10.000 esemplari, contiene una ROM di programma 1344 x 8 e una RAM dati 96 x 4 tutte raggruppate in un solo chip col processore Rockwell PPS-4. Col PPS-4/1 il microcomputer ha 31 canali di Ingresso/Uscita, con doppie possibilità di interruzione.
>
> Secondo il costruttore, la Rockwell Microelectronics Device div., Anaheim, California, con questo microcomputer si possono abbassare i costi dei sistemi elettronici per controlli periferici, apparecchi di controllo e altre applicazioni industriali.
>
> Le opzioni di ingresso/uscita per questo circuito integrato a 50 istruzioni comprendono due canali a 4-bit che possono essere usati contemporaneamente per controllare e confrontare dei dati; due canali di ingresso/uscita a 4-bit e 10 linee discrete di ingresso/uscita. Due linee di ingresso per richiesta di interruzione, una delle quali può automaticamente fornire un trigger a un segnale di eco, forniscono la possibilità di avere ingressi a priorità".

Nell'annuncio c'è dell'altro, ma quello che ci interessa sottolineare è che questo singolo chip è molto più vicino ad un vero *computer su un singolo chip* che ai microprocessori che si trovano oggi sul mercato. Il microprocessor 8080A, di cui abbiamo parlato nel Bugbook V, è un vero e proprio microprocessor; non contiene però memorie a lettura/scrittura, nè ROM, nè possibilità di I/O.

## HARDWARE E SOFTWARE

*Hardware* e *Software* sono due parole molto importanti che verranno usate molto spesso in questo capitolo, quindi è meglio darne subito la definizione:

*Hardware* — I dispositivi meccanici, elettrici, elettronici e magnetici di cui è composto un computer; l'insieme di tutto il materiale che compone un computer.

*Software* — La totalità dei programmi e routine usate per allargare le possibilità dei computer, quali i compilatori, gli assemblatori, le routine e subroutine.

Il microcomputer MMD-1, come ogni altro circuito integrato, cavi e sistemi ausiliari di breadboarding, e dispositivi periferici, sono tutti considerati come hardware. I programmi e le routine che avete usato e scritto, sono invece del software. Avete visto che lo scrivere dei programmi per sfruttare al meglio la memoria disponibile, il set di istruzioni e il tempo richiesto per l'esecuzione di goni singola istruzione, richiede un notevole sforzo.

## COSA È UN CONTROLLORE?

Graf ha definito un *controllore* nel seguente modo:

*Controllore* — Strumento che mantiene un processo o una condizione al livello desiderato o a uno stato stabilito da un confronto del valore reale con quello desiderato.

I controllori possono essere analogici o digitali, elettronici, meccanici, elettromeccanici o pneumatici, o una combinazione di questi. Un *controllore digitale* acquisisce il valore reale della condizione in forma digitale e lo confronta col valore desiderato contenuto all'interno del controllore. Se viene scoperta qualche differenza tra i due, invia un segnale digitale al dispositivo, macchina o processo che dà il via ad una azione che tende a ridurre la differenza. Il controllore digitale consiste di un circuito integrato e di componenti discreti che vengono collegati ad una piastra di circuito stampato, oppure di un computer di qualsiasi grandezza con un limitato numero di chip che servono da interfaccia tra il computer e il mondo esterno.

Il fattore costo diventa importante quando si prende in considerazione l'uso che si fa del controller o del computer. Nessuno penserebbe di controllare 100 dispositivi, ognuno dei quali del valore di 500 dollari, con un computer da 1.000.000 di dollari; usare un computer così grosso per controllare un valore di 50.000 dollari è una specie di suicidio. D'altro canto, un computer di questo tipo potrebbe benissimo essere impiegato per controllare un impianto clinico del valore di 20.000.000 di dollari. Quindi, con la tecnologia disponibile oggi, è difficile che si richieda come controllare un computer da 1 milione di dollari; probabilmente un buon minicomputer da 200.000 dollari è abbastanza per controllare qualsiasi impianto.

Si può giustificare il costo di un computer/controllore se esso rappresenta solo una percentuale minima del costo operativo di un processo o della produzione di un prodotto. I costi e le prestazioni cambiano continuamente così come decrescono i prezzi dei computer. Con l'avvento dei microcomputer, il costo delle apparecchiature di controllo diminuirà certamente senza che si perda in affidabilità.

## DOVE SI COLLOCANO I MICROCOMPUTER

L'articolo di *Businnes Week,* di cui abbiamo preso gli estratti che abbiamo riportato in questo capitolo, fornisce anche delle previsioni sulle potenziali applicazioni del microcomputer. Parleremo più in dettaglio di questo argomento aiutandoci con la figura 16-1 e con la tabella 8.

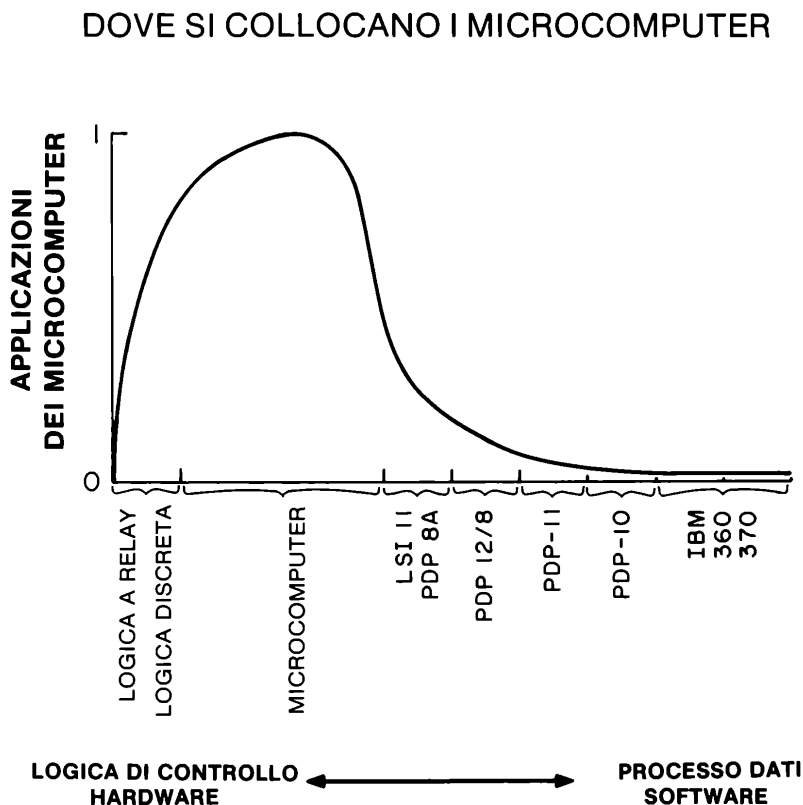



*Figura 16-1.* Diagramma che rappresenta le applicazioni previste dei microcomputer, a metà strada ta la logica discreta e i microcomputer poco costosi.

| LUNGHEZZA DELLA PAROLA | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|
| COMPLESSITA' | Logica cablata | Array logici programmati | Calcolatrice | Microprocessor | Minicomputer | Grande computer | |
| APPLICAZIONE | | Controllo | | Calcoli Elaborazione dedicati | | Dati generali a basso costo | Alte prestazioni nella elaborazione dati generali |
| COSTO | 100$ (1974) | | 1000$ (1974) | | | 10.000$ (1974) | 100.000$ e più (1974) |
| GRANDEZZA DELLA MEMORIA | Molto piccola 0-4 parole | Piccola 2-10 parole | | Media 10-1000 parole | | Grande 1000-1 milione di parole | Molto grande più di 1 milione di parole |
| PROGRAMMI | Solo lettura | | | | | | Rilocabile |
| LIMITI DI VELOCITA' | Tempo reale | Lento | | Medio | | | Orientato al throughput |
| INGRESSO/ USCITA | Integrato | Pochi dispositivi semplici | | Alcuni dispositivi complessi | | | Tutte le apparecchiature |
| PROGETTO | Logico | Logica + microprogrammi | | Micro e macro programmi | | | Macroprogrammi con linguaggi evoluti-sistemi di software |
| VOLUME DI PRODUZIONE | Grande | | | | | | Piccolo |

*Tabella 8.* Spettro della complessità dai computer, dai sistemi a semplice logica cablata ai grandi sistemi a grandi prestazioni per dati di ogni tipo. La tabella è tratta da un articolo di Wallace B. Riley uscito su *Electronics* il 17 ottobre 1974. Nell'articolo si dice che i dati sono presi da materiale della Pro-Log Corporation.

Nella figura 16-1 abbiamo riportato il numero delle applicazioni dei microcomputer in forma di rapporto da 0 a 1, rispetto al tipo di applicazione. Come si può notare, non ci si può aspettare che molti dei microcomputer di oggi vengano usati come macchina calcolatrice o semplicemente per sostituire dei sistemi di relay logici. Tipicamente la maggior parte delle applicazioni significative ricade da una parte, nell'insieme della logica discreta (porte e flip flop) e dall'altra nell'insieme dei microcomputer poco costosi. I microcomputer stanno creandosi dal nulla un nuovo mercato che non è stato mai servito nè dai minicomputer - a causa del loro costo - nè da complessi circuiti digitali. Questo è il regno delle macchine intelligenti, regno che crescerà a spese sia della logica discreta che dei minicomputer man mano che diminuirà il costo dei microcomputer. In questo momento i costi non giustificano la costruzione di minicomputer o di computer più grossi partendo dal chip microcomputer.

Il problema base coi minicomputer è il software; la Data General e la Digital Equipment Corp hanno un vantaggio sulla Intel, la Texas Instruments e la National Semiconductors, e questo vantaggio risiede tutto nel sofisticato software disponibile per il diffusissimo PDP-8, PDP11 e per il NOVA, ma pensiamo che questo vantaggio non durerà ancora per molti anni. È già disponibile una versione del BASIC per i microcomputer 8080 e sta per essere sviuluppato un package APL. I costruttori di minicomputer hanno risposto sviluppando dei microcomputer che hanno compatibilità al software con i minicomputer e l'esempio più significativo è il NOVA. Probabilmente assisteremo alla fusione delle tecnologie dei mini e microcomputer.

Nel punto più alto della gamma dei computer non è possibile rimpiazzare queste grandi macchine come il PDP 10, l'IBM 360 e l'IBM 370 con dei microcomputer. Tuttavia una società californiana ha proposto di usare 256 microcomputer 8080A collegati in forma di "ipercubo"; secondo loro un computer di questo tipo avrebbe potuto rivaleggiare e addirittura prevalere sui grandi computer. Può darsi che i computer delle generazioni future traggano qualche vantaggio dall'architettura di computer distribuiti, ma, ancora una volta, il problema è la produzione del software.

La tabella 8 rappresenta la gamma dei computer suddivisi in funzione della complessità., dai semplici sistemi a logica cablata alle apparecchiature di elaborazione dati ad alte prestazioni. I costi decrescono lungo l'asse. Ogni cinque anni il costo di una quantità equivalente di capacità di elaborazione diminuisce di un fattore 10.

## GERARCHIE DI COMPUTER

Una *gerarchia* è una serie di elementi classificati secondo un rango o un ordine[2]. I microcomputer controlleranno il funzionamento di macchine singole o di strumenti; i minicomputer raccoglieranno i dati da gruppi di microcomputer e li confronteranno con dei modelli matematici molto complessi rappresentanti il processo che viene controllato da circa dieci microcomputer; i grandi computer interrogheranno periodicamente i minicomputer sullo stato dell'intero processo e potranno rielaborare le informazioni raccolte in modo da renderle di facile comprensione ai supervisori della produzione. Nella figura 16-2 rappresentiamo la gerarchia consistente di sette microcomputer basati sull'8080 e su un singolo minicomputer. Le comunicazioni tra il minicomputer e microcomputer potrebbero avvenire anche in modo seriale.

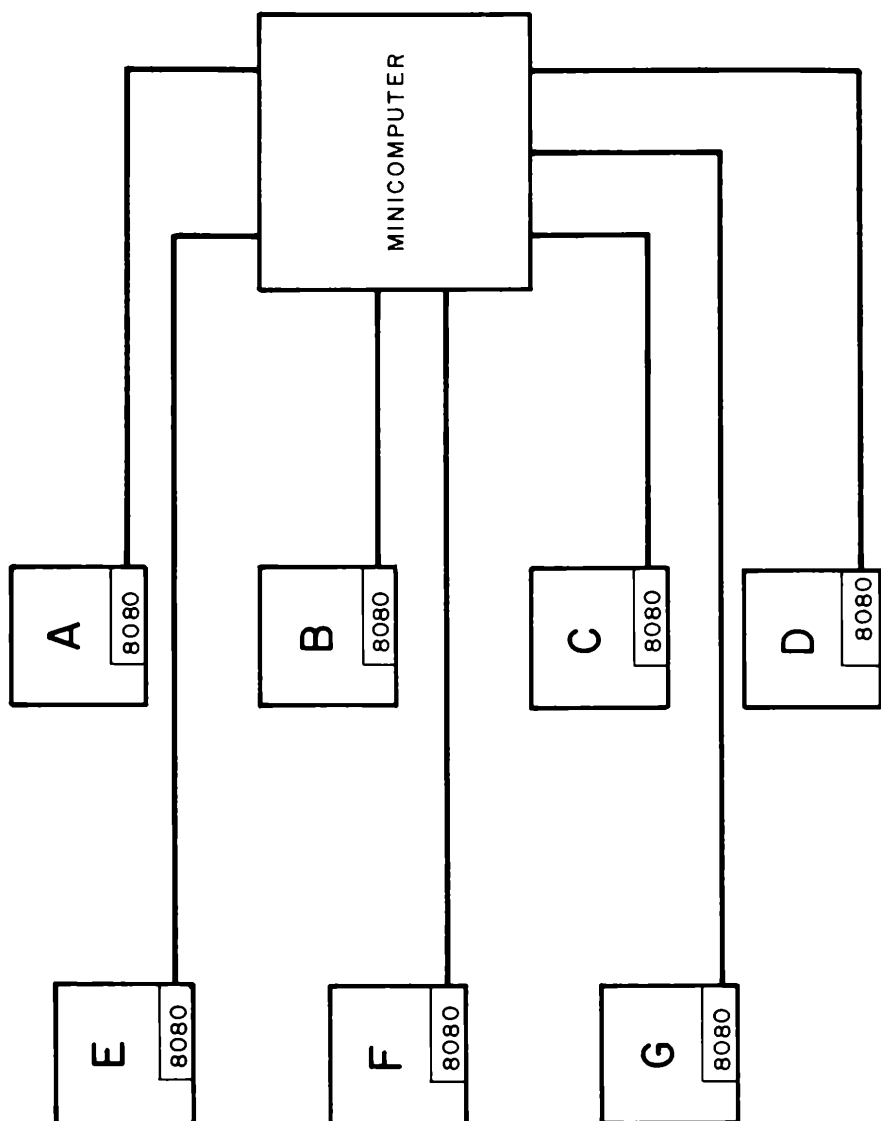
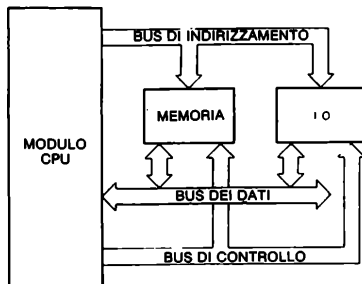



*Figura 16-2.* Esempio di gerarchia di computer. I singoli strumenti da A a G sono controllati da un microprocessor 8080. Inoltre questi microprocessori comunicano in due sensi col microcomputer, che controlla il funzionamento dell'intero sistema.

## UN TIPICO MICROCOMPUTER 8080

Nella figura qui sotto è mostrato un tipico microcomputer costruito partendo da un microprocessor 8080A:

*Courtesy of Intel Corporation, Santa Clara, California*

Diamo adesso alcune definizioni:

*Bus* — Una via attraverso la quale vengono trasferite le informazioni, da una qualsiasi sorgente ad una qualsiasi destinazione. I trasferimenti avvengono uno per volta e quando sta avendo luogo un trasferimento, tutte le altre sorgenti di informazioni collegate, devono essere disabilitate.

*Data bus bidirezionale* — È un bus dei dati attraverso il quale le informazioni possono essere trasferite in entrambe le direzioni. Se ci riferiamo ad un microcomputer basato sull'8080A, il data bus bidirezionale è la via attraverso la quale i dati vengono trasferiti tra CPU, memorie e dispositivi di ingresso/uscita.

*Bus di indirizzamento* — È un bus monodirezionale attraverso il quale una informazione digitale riesce ad identificare una particolare locazione di memorie ad un particolare dispositivo di I/O. Il bus di indirizzamento dell'8080A è un insieme di sedici linee.

*Indirizzo* — È un gruppo di bit che identifica una specifica locazione di memoria o un dispositivo di I/O. Un microcomputer basato sull'8080A usa sedici bit per identificare una specifica locazione di memoria e otto bit per identificare un dispositivo di I/O.

*Controllo* — È quella parte di microcomputer che preleva le istruzioni, nella corretta sequenza, le interpreta e genera gli specifici segnali[14].

*Bus di controllo* — È un set di segnali che regola il funzionamento di un microcomputer, compresi di dispositivi di I/O e la memoria. Di solito funzionano come segnali di "traffico" o di comandi. Possono anche segnalare ai dispositivi di I/O di trasferire o ricevere segnali alla o dalla CPU. Secondo la termonologia della Intel Corporation, un bus di controllo è un set monodirezionale di segnali che indica il tipo di attività - lettura di memoria, scrittura di memoria, lettura di I/O, scrittura di I/O i riconoscimento di un segnale di interruzione - del processo in atto.

| | |
|---|---|
| *I/O* | Abbreviazione di input-output (ingresso-uscita). |
| *Dispositivo I/O* | Dispositivo di ingresso-uscita. È un lettore di schede, una unità a nastro magnetico, una stampante, o un dispositivo di questo tipo che riceve o trasmette dati da o per il computer o dispositivi secondari di memorizzazione. In senso lato, è un qualsiasi dispositivo digitale compreso il singolo circuito integrato, che trasmette o riceve dati o impulsi di strobe da un computer. |
| *CPU* | Abbreviazioni di unità centrale di elaborazione (ingl.: Central Processing Unit). |
| *Unità centrale di elaborazione (grandi computer)* | Detta anche processore centrale. Parte di un sistema di elaborazione che contiene la memoria principale, l'unità aritmetica e i gruppi di registri speciali. Esegue le operazioni aritmetiche, controlla l'esecuzione delle istruzioni e fornisce i segnali di timing e quelli necessari alle operazioni di governo centrale. |
| *Unità centrale di elaborazione (microprocessori)* | Singolo circuito integrato che esegue il trasferimento dati e operazioni di controllo, di ingresso-uscita aritmetiche e logiche, eseguendo le istruzioni che prende dalla memoria. |
| *Memorie* | È un dispositivo che può immagazzinare i bit allo stato logico 0 o 1 in modo tale che un singolo bit, o un gruppo di essi, possa essere ritrovato e letto. |

Un tipico microcomputer costruito su un microprocessor 8080A possiede tutti i requisiti minimi di un computer digitale:

- È programmabile, con dati ed istruzioni di programma che possono essere messi in una qualsivoglia sequenza.
- È digitale
- È sottoposto a logica di clock (nella maggior parte dei microcomputer, le operazioni interne alla CPU avvengono in modo sincrono).
- Contiene l'unità aritmetico/logica, che si trova all'interno del chip, la quale esegue le operazioni aritmetiche e logiche
- Può scambiare dati con la memoria e i dispositivi di I/O.
- Contiene la memoria "veloce"; la velocità è un requisito importante per un computer digitale.

## BUS DI INDIRIZZAMENTO

Il microprocessor 8080A della Intel contiene un bus di indirizzamento a 16 bit che viene usato per identificare una specifica locazione di memoria o uno specifico dispositivo di I/O . È un bus monodirezionale, cioè l'indirizzamento dell'informazione può partire solo dal chip 8080A. Con 16 bit di indirizzamento si possono identificare, e quindi indirizzare, $2^{16}$ = 65.536 diverse locazioni di memoria; diciamo quindi che l'8080A è un dispositivo a "64K" dove "K" è l'abbreviazione di kilobyte, cioè 1024 byte.

Il bus di indirizzamento dell'8080A della Intel, può essere usato anche per fornire il codice di indirizzamento di 8 bit per i dispositivi di I/O. Quando indirizza dei dispositivi di input-output ingresso/uscita, il bus di indirizzamento assume una diversa configuarazione, cioè viene suddiviso in due identici byte di 8 bit di codice di indirizzamento di un dato dispositivo, entrambi i quali possono essere usati per collegare un circuito di interfacciamento con i dispositivi di I/O. Quando si indirizzano i dispositivi di I/O usando le istruzioni IN e OUT, si possono quindi riconoscere $2^8$=256 diversi dispositivi di ingresso e $2^8$=256 diversi dispositivi di uscita.

Ogni volta che si incontra il termine *bus*, bisogna fare attenzione al fatto che diversi tipi di informazioni compaiono sulle linee del bus in momenti diversi. Nel caso del bus di indirizzamento dell'8080A questo è certamente vero. L'informazione che appare più frequentemente sul bus di indirizzamento è l'indirizzo di una specifica locazione di memoria, e solo occasionalmente, appare l'indirizzamento di un dispositivo di I/O. È il microcomputer stesso che sa quando il bus viene usato per accedere alla memoria e quando viene usato per identificare un dispositivo di I/O: *egli stesso fornisce l'appropriato impulso di controllo che vi informa di cosa sta facendo!.* Discuteremo di questo impulso in un altro paragrafo.

## DATA BUS BIDIREZIONALE

Il microprocessor 8080A della Intel contiene una data bus bidirezionale di 8 bit che permette di trasferire 8 bit di dati, cioè un byte, tra il chip 8080A e la memoria o i dispositivi di I/O. In diversi momenti appaiono dati diversi sulle linee del data bus, ma per la maggior parte del tempo ci sono istruzioni che vengono dalla memoria. Altrimenti appare un dato tra i seguenti:

- Byte di dati che viene in ingresso da un dispositivo di ingresso.
- Byte di dati che va in uscita su un dispositivo di uscita.
- Byte di dati letto o da scrivere in memoria.
- Bit di controllo di stato per generare qualche segnale per il bus di controllo.
- Byte di indirizzo HI o LO che va memorizzato in un'area di memoria detta *stack*.
- Byte di indirizzo HI o LO che è stato ripreso dallo stack.
- Byte di istruzione che è stato bloccato da un dispositivo di I/O durante una interruzione.

Come si può fare per sapere quale di questi diversi tipi di dati è in trasferimento? È il microcomputer stesso che lo dice *fornendo un appropriato impulso di stato o di controllo che vi informa sul tipo di attività del processo in corso.* A questo punto è chiaro che bisogna conoscere il bus di controllo per capire il funzionamento del microcomputer 8080A. Questa informazione è vera per ogni tipo di computer digitale che incontrerete.

## BUS DI CONTROLLO

Benché conosciuto come bus di controllo, il set dei segnali di cui stiamo parlando noi è in realtà un vero e proprio bus perché *non* appaiono diversi tipi di informazioni sulla stessa linea in tempi diversi.

Ogni linea di segnale è monodirezionale e monofunzionale. Dopo aver dato questo avvertimento, continueremo a chiamare il set di segnali di controllo del chip 8080A col nome di bus di controllo; questo termine è infatti ormai troppo usato nei testi sui microcomputer perchè sia possibile suggerire un nome alternativo più realistico.

Le cinque attività base in cui può essere impegnato il microprocessor 8080A sono le seguenti:

1. Lettura della memoria (Memory read)
2. Scrittura in memoria (Memory write)
3. Lettura di I/O (I/O read)
4. Scrittura di I/O (I/O write)
5. Interrupt o riconoscimento di interrupt (Interrupt Acknowledge)

Diamo la definizione dei termini più usati:

*Leggere (read)* — Trasmettere dei dati da una specifica locazione di memoria a qualche altro dispositivo digitale.

Sinonimo di riprendere, *ricercare (retrieve)*

*Scrivere (write)* — Trasmettere dei dati da qualche dispositivo digitale ad una specifica locazione di memoria. Sinonimo di memorizzare, immagazzinare *(store).*

*Interrupt* — In un computer digitale è l'arresto della normale esecuzione di un programma eseguita in modo che in seguito il programma possa essere ripreso dal punto dove era stato sospeso.

Ci sono cinque diverse linee per i segnali di controllo, una per ognuna delle attività che abbiamo elencato. Queste linee hanno le seguenti abbreviazioni:

1. Lettura della memoria: MEMR (memory read)
2. Scrittura in memoria: MEMW (memory write)
3. Lettura di I/O: I/O R o IN (I/O read)
4. Scrittura di I/O: I/OW o OUT (I/O write)
5. Riconoscimento di un interrupt: INTA o I ACK (Interrupt Acknowledge)

È da notare che, in ogni caso, il segnale è un impulso di clock negativo.

La durata dell'impulso dipende dalla velocità del microcomputer basato sull'8080A; nel microcomputer MMD-1, che ha un clock di 750 kHz, la durata dell'impulso è di 1,333 $\mu$s. L'unicità dei segnali di controllo può essere meglio compreso con l'aiuto della tabella della verità che riportiamo nella pagina seguente. Questi segnali di controllo sono disponibili sul bus della piastra SK10 dell'MMD-1 (figura 16-3).
Di solito li userete (specialmente $\overline{IN}$ e $\overline{OUT}$), per attuare il gate del trasferimento dei dati tra i circuiti integrati (collegati sul breadboard) e la CPU del microcomputer.

| $\overline{\text{MEMR}}$ | $\overline{\text{MEMW}}$ | $\overline{\text{IN}}$ | $\overline{\text{OUT}}$ | $\overline{\text{IACK}}$ | Operazione |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | Legge un byte dalla memoria |
| 1 | 0 | 1 | 1 | 1 | Scrive un byte in memoria |
| 1 | 1 | 0 | 1 | 1 | Legge un byte da un dispositivo di I/O |
| 1 | 1 | 1 | 0 | 1 | Scrive un byte su un dispositivo di I/O |
| 1 | 1 | 1 | 1 | 0 | Attua lo strobe di un byte nel registro istruzioni durante un interrupt o un riconoscimento di interrupt |

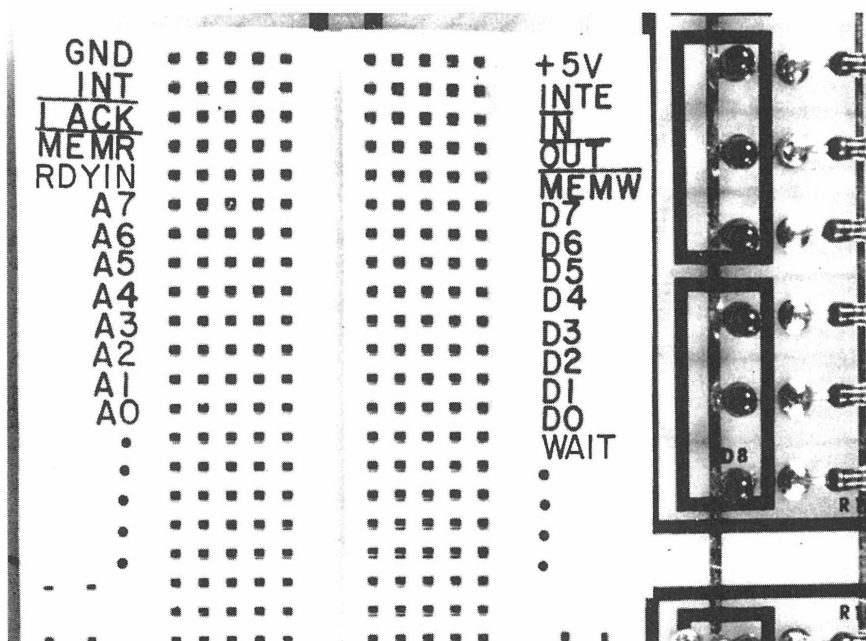



*Figura 16-13* Segnali disponibili sul microcomputer MMD-1. Da A0 ad A7 ci sono gli otto bit meno significativi del bus di indirizzamento; da D0 a D7 ci sono i data bus bidirezionali; INTE è l'uscita del flip flop abilitatore dell'interrupt; INT è l'ingresso della richiesta di interrupt; $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IN}}$, $\overline{\text{OUT}}$ e $\overline{\text{IACK}}$ sono le uscite dei segnali di controllo. RDYIN e WAIT vengono usati col circuito passo-passo descritto nel Bugbook V capitolo n. 11 esperimento n° 5.

## COSA VUOL DIRE INTERFACCIARE?

*Interfacciare* può essere definito come il collegare elementi di un gruppo (persone, strumenti, ecc.) in modo che essi siano in grado di funzionare in modo compatibile e coordinato[17]. Per "modo compatibile e coordinato" di solito, intendiamo sincronizzato. Diamo adesso alcune importanti definizioni.

| | |
|---|---|
| *Sincrono* | In fase o al passo, quando riferito a due dispositivi o macchine; quando invece si riferisce ad un computer, si intende che la sequenza delle operazioni è controllata da impulsi o segnali di clock. |
| *Computer sincrono* | Computer digitale in cui tutte le operazioni ordinarie vengono sottoposte ad un clock unico. |
| *Operazione sincrona* | Operazione di un sistema sotto il controllo di impulsi di clock. |
| *Logica sincrona* | Tipo di logica digitale usata in un sistema in cui le operazioni logiche avvengono in sincronia con gli impulsi di clock. |
| *Sync* | Abbreviazione di sincrono (ingl. synchronous), sincronizzazione, sincronizzato. |
| *Sincronizzare* | Costringere un elemento di un sistema a stare al passo con un altro. |
| *Impulsi di sincronizzazione* | Impulsi originali dall'equipaggiamento di trasmissione e immessi in un equipaggiamento ricevente per costringere i due equipaggiamenti a funzionare al passo. |
| *Ingressi sincroni* | Quegli ingressi del flip-flop che non controllano direttamente l'uscita, come quelli delle porte, ma lo fanno solo quando il clock lo permette o lo comanda. |

Tutte queste definizioni sono state ricavate dal riferimento 2 dell'appendice 1. Adesso possiamo definire l'*interfacciamento di un computer* come:

| | |
|---|---|
| *Interfacciamento di un computer* | Sincronizzazione della trasmissione dei dati digitali tra un computer ed i dispositivi esterni, comprese le memorie e i dispositivi di I/O. |

Benché i dettagli dell'interfacciamento di un computer varino col tipo di computer impiegato, i principi generali dell'interfacciamento si applicano ad un grande numero di computer. Nella figura 16-4 sono riassunti i punti base dell'interfacciamento del microcomputer 8080A. Se volete interfacciare il microcomputer, dovete:

- Sincronizzare il trasferimento di 8 bit di dati tra il microcomputer e ogni dispositivo di uscita.
- Sincronizzare il trasferimento di 8 bit di dati tra ogni dispositivo di ingresso e il microcomputer.
- Generare gli appropriati impulsi di sincronizzazione del trasferimento dati in ingresso e uscita, questi sono detti *impulsi di selezione di dispositivo.* Per un microcomputer basato sull'8080A, si possono generare 256 diversi impulsi di sincronizzazione di ingresso e 256 diversi impulsi di sincronizzazione di uscita.

- Fornire segnali di interrupt che arrivi al microcomputer da dispositivi di I/O esterni.
- Programmare il microcomputer perchè esegua tutte le operazioni di ingresso-uscita e i servizi di interrupt.

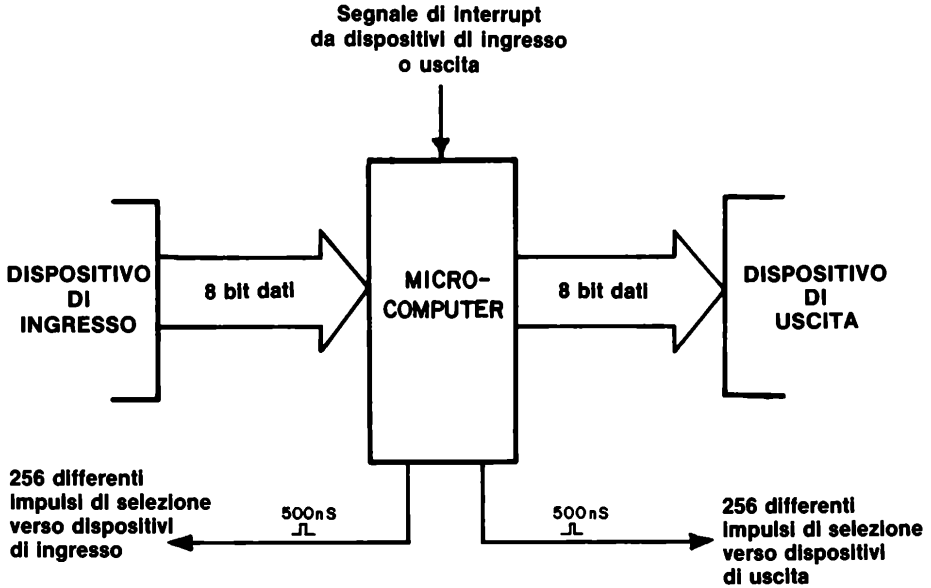
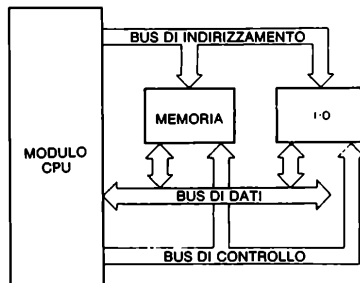



*Figura 16-4.* I quattro principali compiti dell'interfacciamento: ingresso, uscita, generazione dell'impulso di selezione dispositivo, e servizio di interrupt.

Nello schema seguente riportiamo un modo migliore di illustrare tre dei quattro compiti dell'interfacciamento.

Il trasferimento di 8 bit di dati tra la CPU e i dispositivi di I/O avviene sul bus bidirezionale di 8 bit. Lo specifico dispositivo di I/O interessato al trasferimenti dei dati, viene selezionato mediante 8 bit del *bus* di indirizzamento.

La temporizzazione del trasferimento dati viene determinata dalla presenza di un impulsi $\overline{IN}$ o $\overline{OUT}$ sul bus di controllo. Come si vede, al trasferimento di dati tra la CPU e un dispositivo di I/O, *sono interessati tutti e tre i bus.*

Ci sarebbe ancora molto da dire sull'interfacciamento, ma ne parleremo nei prossimi capitoli.

## COSA È UN DISPOSITIVO DI I/O?

Due delle definizioni più usate sono:

*Ingresso-uscita (I/O)* — Termine generale per la strumentazione usato per comunicare con un computer e per i dati coinvolti nella comunicazione.

*Dispositivo di I/O* — Ogni dispositivo digitale, compreso il singolo circuito integrato, che trasmette, o riceve dati o impulsi di strobe da un computer.

Il punto di vista tradizionale per un dispositivo di I/O è qualcosa di più complesso; lettori di schede, unità a nastro magnetico, video e telescriventi ricadono sotto questa definizione, mentre anche un singolo circuito integrato, come un latch, un buffer three-state, un registro shift, un contatore, o una piccola memoria, possono essere considerati dispositivi di I/O.

Abbiamo detto poco fa che bisognava sincronizzare il trasferimento dei dati tra un computer e un dispositivo di I/O, e che questa sincronizzazione viene eseguita mediante l'uso di impulsi detti *impulsi di selezione di dispositivo.* È importante notare che un singolo dispositivo di I/O può avere bisogno di più impulsi di selezione dispositivo. Per esempio, il registro di shift 74198 ha una coppia di ingressi di controllo che determinano se il registro deve fare uno shift a destra, a sinistra o caricare otto bit di dati in parallelo. Inoltre ha anche un ingresso di clear e un ingresso di clock, quindi un singolo chip 74198 può avere bisogno di tre o quattro impulsi di selezione dispositivo diversi, e quindi, il fatto che si possano generare 256 diversi impulsi di selezione dispositivi di uscita e 256 diversi impulsi di selezione dispositivi di uscita e 256 diversi impulsi di selezione dispositivi di ingresso, non vuol dire che si possono indirizzare 512 "dispositivi" diversi. In effetti si riescono a indirizzare 50 o 100 dispositivi, esistono altri modi per generare questi impulsi.

Nei capitoli successivi, imparerete a generare gli impulsi di selezione dispositivo, e parleremo anche dei vari usi di questi impulsi.

## DOMANDE RIEPILOGATIVE

Le seguenti domande servono ad aiutarvi a ripassare i più importanti concetti dell'interfacciamento dei microcomputer.

1. Quali dei seguenti elementi sono hardware e quali sono software?

    a. Cross assembler

    b. Editor

    c. Circuito integrato

    d. Filo o cavo

    e. Scheda di circuito stampato

    f. Condensatori e resistori

    g. Programma FORTRAN

    h. Termistore (trasduttore di temperatura)

2. Quali sono i tre bus più importanti di un microcomputer?

3. Perchè il bus dati è di solito bidirezionale?

4. Elencate i cinque più importanti segnali di controllo in un microcomputer basato sull'8080A

**RISPOSTE**

1. a. Software
   b. Software
   c. Hardware
   d. Hardware
   e. Hardware
   f. Hardware
   g. Software
   h. Hardware

2. Il data bus bidirezionale, il bus di controllo e il bus di indirizzamento.

3. Perchè per un microprocessor di 8 bit, riduce il numero dei pin a 8, mentre un microprocessor a 16 bit, riduce il numero dei pin a 16.

4. $\overline{OUT}$, $\overline{IN}$, $\overline{MEMR}$, $\overline{MEMW}$ e $\overline{IACK}$

# CAPITOLO 17

# IMPULSI DI SELEZIONE DISPOSITIVO

## INTRODUZIONE

Questo capitolo vi insegnerà come generare gli impulsi per selezionare dispositivi di ingresso/uscita, cioè quei segnali generati dal microcomputer, che servono a sincronizzare il microcomputer stesso e i dispositivi esterni di ingresso/uscita: tali dispositivi possono essere dei semplici circuiti integrati. Il circuito più usato è un circuito che si basa sul chip decodificatore T4154; possono essere generati sedici diversi impulsi di selezione dispositivo.

## OBIETTIVI

Alla fine di questo capitolo sarete in grado di:

- Dare la definizione di impulsi di selezione dispositivo
- Spiegare che cosa si intende dire con l'espressione "la sostituzione dell'hardware con il software"
- Fare un elenco dei diversi impieghi degli impulsi di selezione dispositivo
- Dare uno o due schemi di circuiti che possono essere usati per generare impulsi di selezione dispositivo
- Dare la definizione di "stato" e di "ciclo macchina"
- Elencare quanti cicli macchina esistono per alcune istruzioni tipiche dell'8080 A
- Cablare un circuito che vi permetterà di far avanzare di un passo il microcomputer MMD-1

## CHE COSA È UN IMPULSO DI SELEZIONE DISPOSITIVO

Un impulso di selezione dispositivo (Device Select Pulse) è un impulso di sincronizzazione generato da un calcolatore digitale per sincronizzare il trasferimento dei dati fra il calcolatore e il dispositivo di ingresso-uscita. Associate il termine "impulso di selezione dispositivo" con i termini abilitare, registrare, collegare, disabilitare, inibire e temporizzare. Fondamentalmente, un impulso di selezione dispositivo è un impulso che attiva lo Strobe di alcune operazioni in un circuito digitale o integrato. Esso può essere sia un impulso di gate - impulso che abilita un circuito di gate al passaggio di un segnale digitale - che un impulso clock (o di temporizzazione) in un sistema "clocked logic".

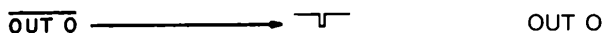
## LA SOSTITUZIONE DELL'HARDWARE CON IL SOFTWARE:

## USO DELL'IMPULSO DI SELEZIONE DISPOSITIVO

Gli impulsi di selezione dispositivo sono facili da generare e non sono costosi. Generalmente, viene usato un chip decodificatore 74L154.

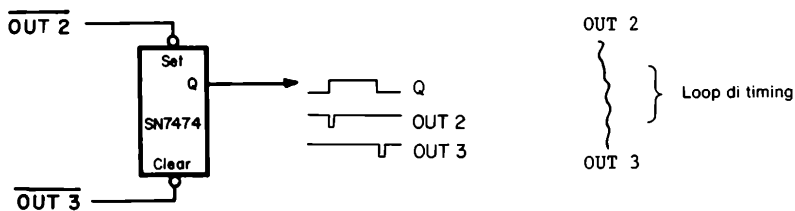
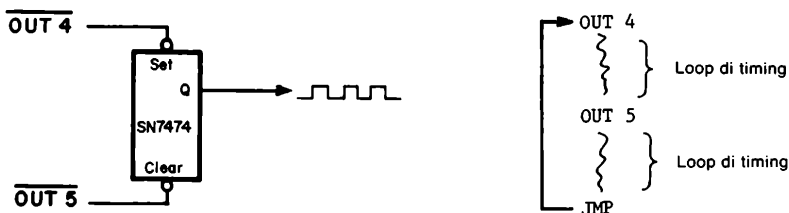
Nell'interfacciare un microcomputer, il vostro scopo è quello di minimizzare il numero di chip esterni richiesti, supponendo che costruiate centinaia di unità e non una unità per ogni tipo. Un modo di minimizzare i chip esterni è quello di produrre la logica digitale all'interno del microcomputer invece che all'esterno.
Noi chiamiamo questo procedimento *la sostituzione dell'hardware con il software.* Ricordate questo concetto: software al posto dell'hardware. Il vostro scopo principale, nell'uso dei microcomputer, e di sostituire i dispositivi hardware elettronici e meccanici con i programmi del microcomputer stesso. Dato che, nei microcomputer classici, molti dispositivi di ingresso-uscita sono lenti, otterrete molto successo nei casi in cui avrete a che fare con tali dispositivi. Comunque, può capitarvi di trovarvi in una situazione in cui la velocità del microcomputer non è abbastanza elevata. Ci vuole tempo per eseguire tutte le istruzioni del programma di un microcomputer. Se il programma è troppo lungo, verrà impiegato troppo tempo e potreste non essere in grado di assolvere un compito specifico.

Per dimostrare come sostituire l'hardware con il software, vorremmo illustrarvi diversi modi d'impiego degli impulsi di selezione dispositivo . In ogni esempio presentato, ricordate per favore che esiste associato un programma del microcomputer che dà il tempo di generazione degli impulsi di selezione dispositivo.

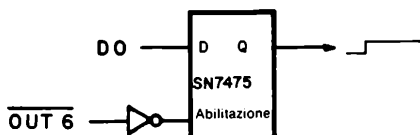
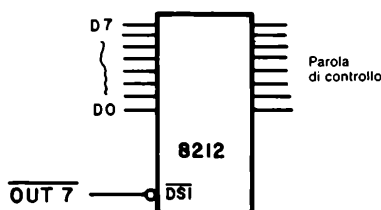
È semplice scrivere un programma che genera un solo impulso di selezione, come l'impulso negativo, $\overline{\text{OUT O}}$, che segue:

Per il microcomputer MMD-1, la lunghezza dell'impulso è di 1,333 microsecondi. In un capitolo successivo imparerete come scrivere vari tipi di loop di delay che eseguono in modo ripetitivo un piccolo gruppo di istruzioni. Scrivendo un programma di questo tipo, sarete in grado di generare una serie di impulsi di selezione dispositivo,

come mostriamo per $\overline{\text{OUT 1}}$,

La durata tra uno e l'altro degli impulsi successivi è determinata da un *loop di delay programmato e generato dal software.*

Con un paio di impulsi di selezione dispositivo ed un solo flip-flop con preset e clear come il 7474, potete scrivere un loop di delay che genera un solo impulso di clock all'uscita del latch,

Aggiungendo al vostro programma un secondo loop di delay potete generare una serie di impulsi di clock con un duty-cycle noto, che è specificato dal programma,

Con questi ultimi due circuiti, avete sostituito con il software sia un multivibratore bistabile che un multivibratore astabile come il timer 555 cablato come un oscillatore. Avete già usato un certo numero di Outboard, come il pulser Outboard ed il clock Ourboard. Con la vostra abilità di sostituire l'hardware con il software, siete ora in grado di rimpiazzare tali Outboard con un chip 7474 (che contiene due flip-flop) e con programmi appropriati del microcomputer.

Nelle piastre di circuito stampato complesse, incontrerete spesso flip-flop e gate che vengono usati per dare impulsi e stati logici in determinati punti all'interno del circuito, e in determinati istanti di tempo. Con il microcomputer, è relativamente facile svolgere compiti di questo tipo.

Potete anche usare un solo bit D0, sul bus di dati bidirezionale, e un impulso di selezione dispositivo $\overline{\text{OUT 006}}$, per "bloccare" il bit e controllare così lo stato logico in un punto particolare del circuito digitale esterno.

Un circuito di interfaccia molto più efficiente è basato su di un latch di 8 bit e un solo impulso di selezione dispositivo,

Con un impulsi di selezione $\overline{\text{OUT 007}}$ e un 8212, è possibile attuare il latch di tutti gli otto bit sul bus di dati bidirezionali ed usare tali bit come linee di controllo individuali in vari punti di un circuito digitale. Nel prossimo capitolo imparerete come realizzare un circuito a latch di questo tipo.

Non sarete in grado di sostituire con il software determinati tipi di chip, come latch e buffer three state ciononostante, se non è critica la velocità, potete usare il software per sostituire la maggior parte delle funzioni MSI importanti e molti circuiti integrati LSI. Questo punto è svolto molto bene, con il microcomputer PACE, nel manuale "Logic Designers Guide to Programmed Equivalents to TTL Functions", che è messo a disposizione dalla National Semiconductor Corporation. Tutti i programmi PACE possono essere convertiti in operazione su di un microcomputer a 8 bit, come l'8080 A o il Motorola 6800. L'hardware che è stato sostuito con il software del microcomputer equivalente comprende:

- 7408 porta AND a 2 ingressi quadrupli
- 7409 porta AND a 2 ingressi quadrupli con uscite open collector
- 7411 porta AND a 3 ingressi tripli collegate insieme
- 74H21 porta AND a 4 ingressi doppi
- 7432 porta OR a 2 ingressi quadrupli
- 7486 porta OR Esclusivo a 2 ingressi quadrupli
- 7483 full-adder binario a 4 bit
- 74121 e 555 multivibratori monostabili (delay maggiori di 10 $\mu$s)
- 74150 selettore/multiplexer dati da 16 a 1 linea
- 74151 selettore/multiplexer dati da 8 a 1 linea

- 74154 decodificatore/demultiplexer da 4 a 16 linee
- DM82220 generatore/controllore di parità da 9 bit
- 7485 confrontatore di grandezza assoluta a 4 bit
- da 74160 a 74163, contatori a 4 bit sincroni, BCD e binari
- 74185 convertitore da binario a BCD
- 74184 convertitore da BCD a binario
- 74190 contatore BCD
- 74191 contatore binario

Sono stati forniti i software equivalenti anche per i seguenti tipi di sistemi digitali, ognuno dei quali consiste di un numero di circuiti integrati della serie 7400:

- (74193, 7485 e varie porte) servomeccanismo digitale
- (74163, 7485, 74123, 7475, 7400, e 7404) tacometro digitale
- (74163, 7485 e 7402) divisore modulo N
- (divisore modulo N, 74160, 7404, e 7400) clock a tempo reale e temporizzatore di intervalli
- (74C14) generazione di numeri pseudo-random
- (DM 8551, 7473 e varie porte) sequenziatore di stati

Benchè sia possibile trattare un microcomputer come un minicomputer o usarlo come un calcolatore superprogrammabile, l'uso prevalente dei microcomputer sarà nei sistemi digitali, nei quali il software sostituisce la maggior parte dell'hardware originariamente presente. Se imparate come sostituire l'hardware con il software, avrete imparato uno degli aspetti più importanti delle applicazioni dei microcomputer.

## USO DEGLI IMPULSI DI SELEZIONE DISPOSITIVO PER LO STROBE DEI CIRCUITI INTEGRATI

Un'applicazione importante per i microcomputer è produrre segnali di alimentazione per operazioni di singoli circuiti integrati in strumenti e dispositivi elettronici. Per esempio, tali impulsi possono:

- Azzerare contatori, registri di shift, flip-flop e latch
- Caricare contatori, registri di shift e latch
- Abilitare multiplexer, demultiplexer, decodificatori, selettori dati, contatori, registri di shift, memorie, codificatori di priorità, UART, e vari altri chip
- Inibire ingressi di clock di contatori e registri di shift
- Settare, azzerare, cambiare di posizione e temporizzare i flip-flop.
- Selezionare lo shift a sinistra, a destra, caricare e inibire funzioni nei registri di shift.

Usando impulsi di selezione dispositivo per controllare l'operazione di singoli circuiti integrati, voi sostituite l'hardware con il software.

Avete già familiarità con alcuni circuiti integrati della categoria MSI. Alcuni di questi hanno bisogno di segnali di strobe o di abilitazione allo scopo di eseguire le loro funzioni digitali. Così:

Contatore a decade 7490:
Un livello logico 1 ai pin 2 e 3 azzera il contatore
Un livello logico 1 ai pin 6 e 7 setta il contatore a 9
L'ingresso clock è al pin 14

Contatore binario 7493:
Un livello logico 1 ai pin 2 e 3 azzera il contatore
L'ingresso di clock à al pin 14

Decodificatore 7442:
Un livello logico 0 al pin 12 abilita l'operazione del decodificatore ottale

Decodificatore 74154:
Un livello logico 0 ai pin 18 e 19 abilita il decodificatore

Flip-flop 7474:
Un livello logico 0 al pin 1 azzera il primo flip-flop
Un livello logico 0 al pin 4 setta il primo flip-flop
L'ingresso D del primo flip-flop è al pin 2
L'ingresso di clock del primo flip-flop è al pin 3

7475 Latch:
Un livello logico 1 al pin 3 abilita i primi due latch
Un livello logico 1 al pin 13 abilita i secondi due latch

Nei sistemi a microcomputer, i decodificatori 7442 e 74154 vengono usati per aiutare a generare gli impulsi di selezione dispositivo. Comunque, possono essere usati anche come decodificatori generali che vengono abilitati da tali impulsi.

### COME SI GENERANO GLI IMPULSI DI SELEZIONE DISPOSITIVO

Per generare un impulso di selezione dispositivo, sono necessari due tipi di informazione dal microcomputer 8080A:

1. Il codice di identificazione a 8 bit, chiamato *codice dispositivo,* del dispositivo di ingresso/uscita

2. Un impulso di sincronizzazione ad un solo bit, o $\overline{IN}$ o $\overline{OUT}$, che sincronizza la decodificazione del codice dispositivo.

L'origine di entrambi i tipi di informazione sta nel software, cioè nelle istruzioni IN e OUT che avete incontrato all'inizio di questo Bugbook. Queste istruzioni comprendono il codice dispositivo a 8 bit e fanno sì che il microcomputer generi l'impulso di sincronizzazione $\overline{IN}$ o $\overline{OUT}$ appropriato. La posizione delle istruzioni IN e OUT nel programma determina l'istante specifico in cui vengono generati gli impulsi di selezione dispositivo.

In altre parole, mentre viene generato un impulso di selezione dispositivo, sia il bus d'indirizzo che il bus di controllo sono attivi. Come mostra la Figura 17-1, il codice dispositivo a 8 bit si ottiene dal bus d'indirizzo a 16 bit, e i due impulsi di sincronizzazione, IN e OUT, dal bus di controllo. Nel microcomputer 8080A, il bus d'indirizzo a 16 bit è suddiviso in due codici dispositivo a 8 bit. Per il chip 8080A della Intel, i due codici sono identici durante l'esecuzione del terzo ciclo macchina $\overline{IN}$ o $\overline{OUT}$.

Che cosa fare del bus d'indirizzo a 8 bit e degli impulsi di sincronizzazione? Potete collegarli al decodificatore da 4 a 16 linee 74154 come mostra la Figura 17-2. Con un solo decodificatore 74154, usate solo quattro degli otto bit del bus d'indirizzo e o $\overline{IN}$ o $\overline{OUT}$. In Figura 17-3, è mostrato un circuito decodificatore di impulsi di selezione dispositivo più avanzato. Vengono usati tutti gli otto bit d'indirizzo, e 17 decodificatori 74154 danno l'opportunità di generare 256 impulsi singoli. Per generare tutti i 512 impulsi di selezione dispositivo di ingresso e di uscita, sarebbero necessari due circuiti del tipo mostrato nella Figura 17-3. Ma questo accade raramente nelle attuali applicazioni delle interfacce.

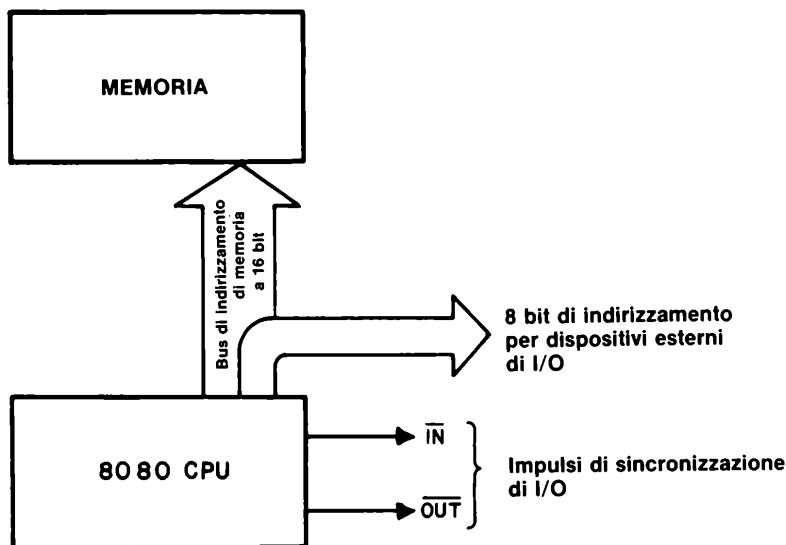



*Figura 17-1.* Per generare un gruppo di impulsi di selezione dispositivo occorrono otto bit dal bus d'indirizzo e due impulsi di sincronizzazione, $\overline{IN}$ e $\overline{OUT}$, dal bus di controllo.

*Figura 17-2.* Per generare sedici diversi impulsi di selezione dispositivo occorrono quattro bit dal bus d'indirizzo e l'impulso di sincronizzazione o $\overline{IN}$ o $\overline{OUT}$, rispettivamente per i dispositivi d'ingresso e di uscita.

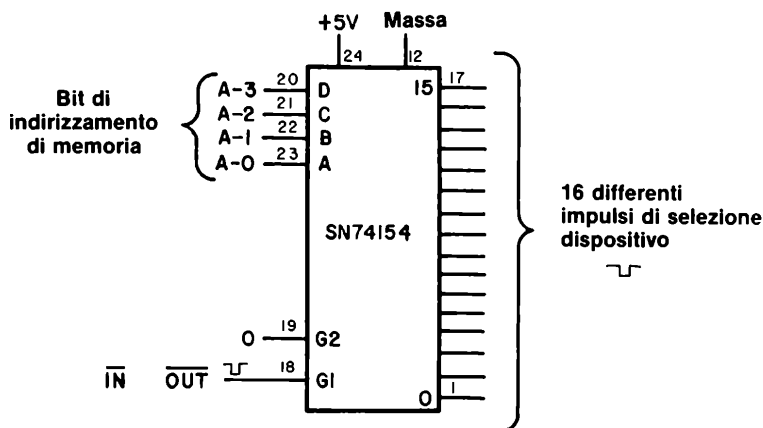
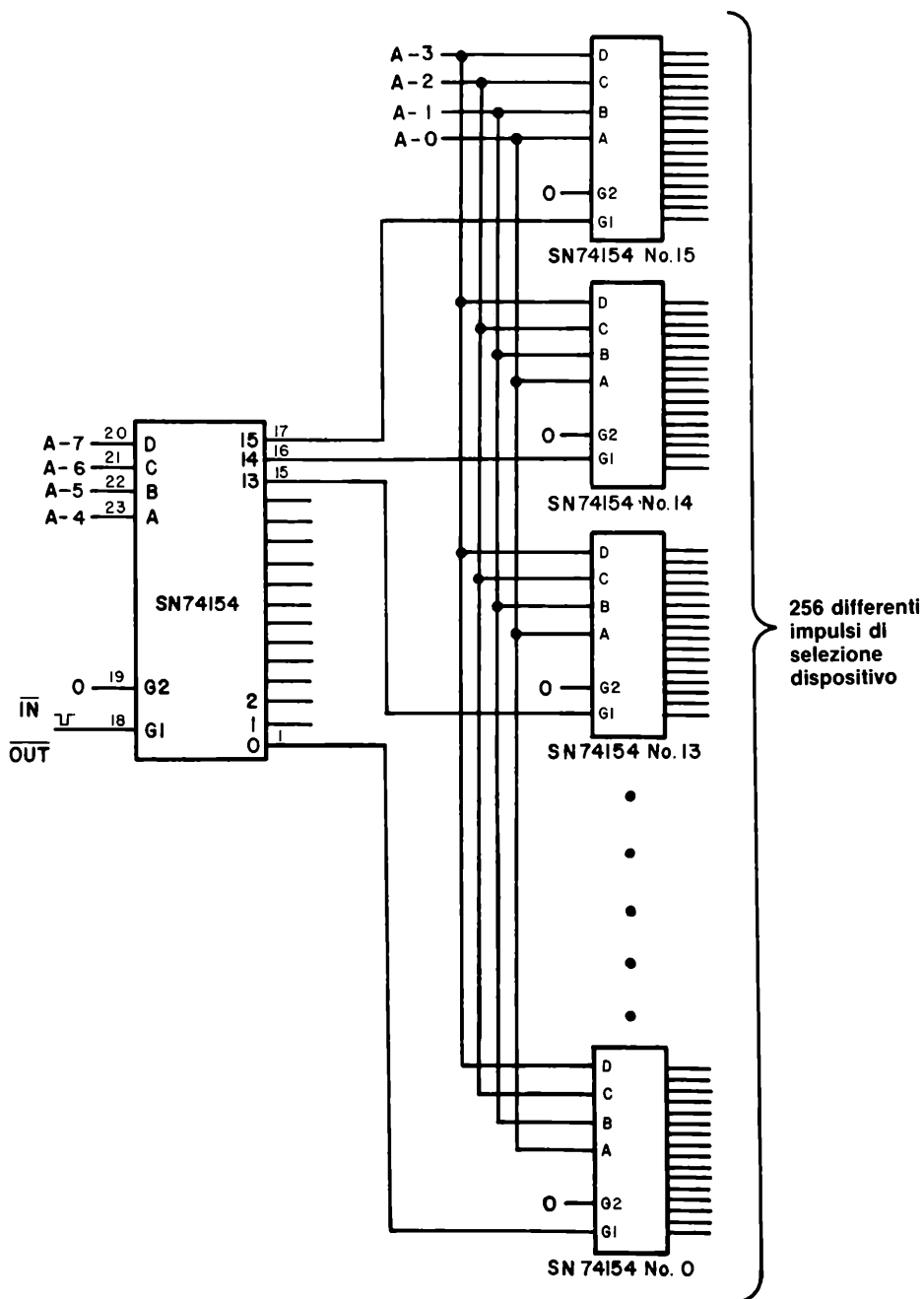



*Figura 17-3.* Circuito per generare 256 diversi impulsi di selezione dispositivo.

Per generare 512 diversi impulsi di selezione dispositivo, userete un sistema differente. Nella Figura seguente, è illustrato un circuito di decodificazione più limitato, basato sulle Figure 17-2 e 17-3:

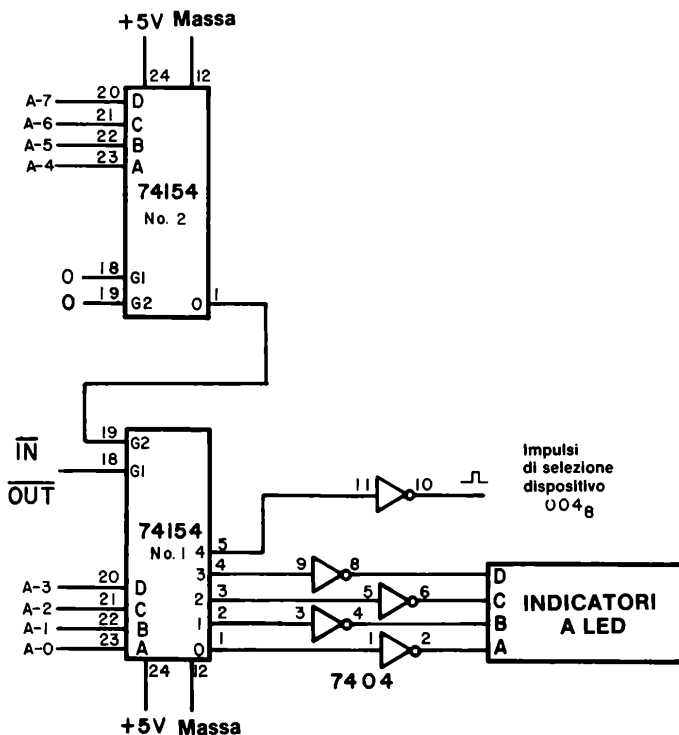



*Figura 17-4.* Un possibile circuito di decodifica per generare sedici impulsi di selezione decodificati in assoluto.

Il circuito comprende una *decodifica in assoluto* di tutto il byte d'indirizzo di selezione dispositivo, cioè tutti gli otto bit, non solo quattro. Questo non è un circuito molto usato: abbiamo voluto soltanto illustrarvelo. Può risultare più utile usare uno dei chip 74154 per gli impulsi di selezione dispositivo d'ingresso, e l'altro per gli impulsi di uscita.

I circuiti visti finora sono quelli che usiamo per generare impulsi di selezione dispositivo. Comunque, esistono schemi alternativi per decodificare i bus d'indirizzo e di controllo, e vorremmo illustrarveli. Nella Figura 17-5, usiamo un paio di decodificatori 74154, e decodifichiamo così in modo assoluto il byte del bus d'indirizzo a 8 bit. Ogni impulso di selezione dispositivo che generiamo richiede una porta 7402 (o 7432), come mostra la Figura 17-6. Questo circuito è utile solo nel caso che gli impulsi di selezione dispositivo del vostro sistema siano distribuiti in maniera casuale nella fascia di valori da 000 a $377_8$ e tutte le funzioni siano centralizzate. Sono necessari solo due chip decodificatori 74154 e quattro porte NOR a 2 ingressi 7402- che contengono in tutto, sedici porte NOR - per ottenere

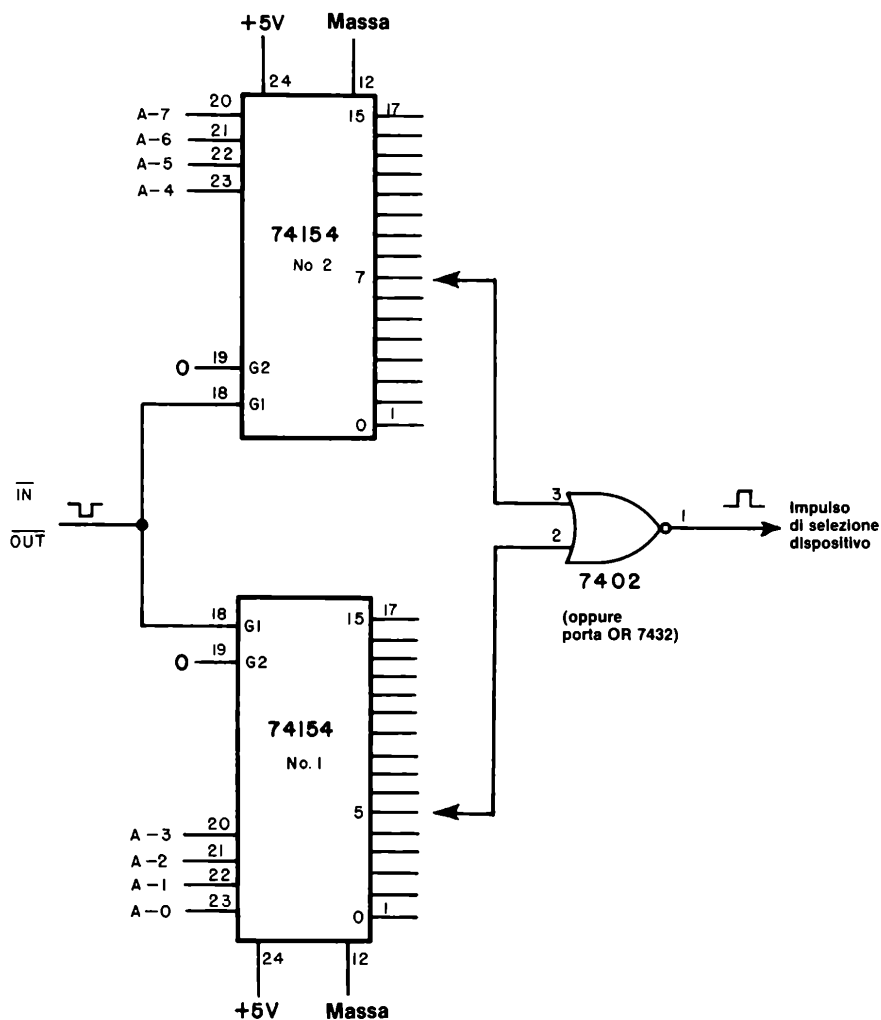



*Figura 17-5.* Schema di decodifica in assoluto per impulsi di selezione dispositivo che richiede una porta 7402 o 7432 per ogni impulso di selezione dispositivo desiderato. Questo non è un circuito molto utile.

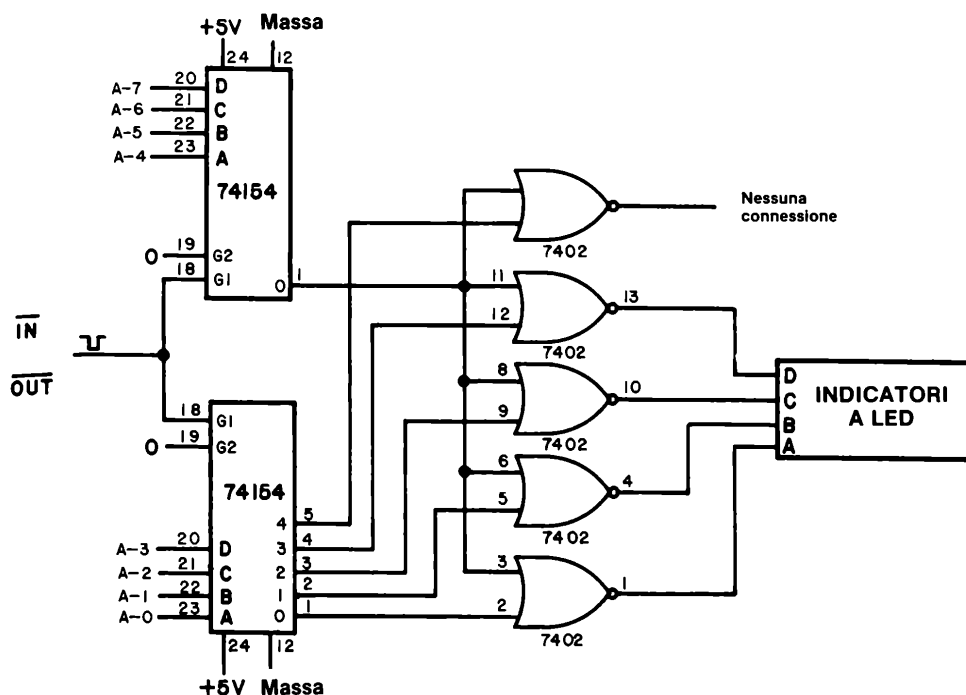



*Figura 17-6.* Circuito che dimostra come vengono usati due porte NOR a 2 ingressi per generare impulsi di selezione dispositivo individuali.

sedici impulsi di selezione dispositivo. Se i codici dispositivo sono sequenziali, questo circuito non viene usato.

Nella Figura 17-7, è mostrata una tecnica di decodifica in relativo. Invece di collegare i segnali di controllo $\overline{IN}$ o $\overline{OUT}$ ad un decodificatore, li collegate a singole porte 7402 (o 7432). Uno schema di decodifica di questo tipo viene usato nel microcomputer MMD-1 (Figura 17-7). Quattro bit del bus d'indirizzo vengono collegati al decodificatore 74L42, i cui canali di uscita sono collegati a loro volta con le porte NOR a 2 ingressi 7402 per fornire gli impulsi di selezione dispositivo necessari ai latch 7475 sulla piastra del microcomputer. Notate nella Figura 17-7 che i bit del bus d'indirizzo da A3 ad A7 sono in collegamento con gli invertitori 74LS05, le cui uscite sono tutte collegate insieme e poi connesse all'ingresso D del decodificatore 74L42.

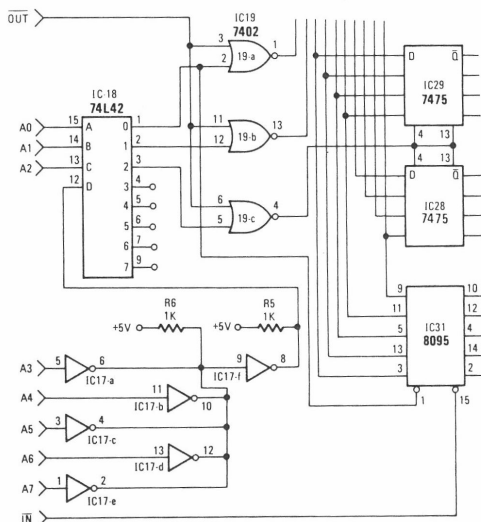



*Figura 17-7.* Circuito di decodifica nel microcomputer MMD-1. Si usa un decodificatore 74L42 per generare impulsi di uscita dei singoli canali, che sono poi collegati alle porte NOR a 2 ingressi 7402 per produrre gli impulsi di selezione dispositivo desiderati.

L'ingresso D deve essere sempre a livello logico 0 se si vogliono generare impulsi di selezione dispositivo. La tecnica qui impiegata è la cosiddetta tecnica *a collettore aperto,* che consiste nel collegare le uscite di speciali chip a collettore aperto, ad un bus comune. Abbiamo in sostanza costruito una porta OR a cinque ingressi, che abilita il decodificatore 74L42. Ne parleremo più avanti.

È possibile anche decodificare dei byte ad 8 bit di codice dispositivo usando porte e comparatori. Ciò risulta particolarmente utile nelle situazioni reali, quando occorrono solo pochi impulsi di selezione dispositivo. Per esempio, prendiamo in considerazione il circuito mostrato in Figura 17-8. A sinistra, vengono generati due impulsi di selezione dispositivo distinti con l'aiuto di due porte NAND a 8 ingressi 7430. Il codice dispositivo per la porta 7430 in alto è $11000110_2$, o 306 in codice ottale, mentre il codice dispositivo per la porta in basso è $11001111_2$, o 317 in codice ottale. L'unica condizione di uscita della porta NAND 7430 è il livello logico 0, che si produce solo quando alla porta è stato applicato il codice dispositivo appropriato. Se anche il segnale di controllo $\overline{\text{OUT}}$ è a livello logico 0, l'unica uscita della porta OR a 2 ingressi 7432, cioè il livello logico 0, o azzera o presetta il flip-flop 7476.

Il circuito della Figura 17-8 dimostra come si possono usare impulsi di selezione dispositivo per controllare una alimentazione c.a. Relè a stato solido e ad isolamento ottico permettono di usare segnali digitali per attivare e disattivare il carico di corrente alternata, operando sia a 115 V che a 220 V. Si possono ottenere dei relè che commuteranno 10 A a questi livelli di tensione. Questo argomento è stato trattato molto più dettagliatamente nel Bugbook III, Capitolo 5.

Notate che i codici dispositivo corrispondono ai codici ASCII a 8 bit per i caratteri "F" e "O", nei quali il bit di parità è a livello logico 1.

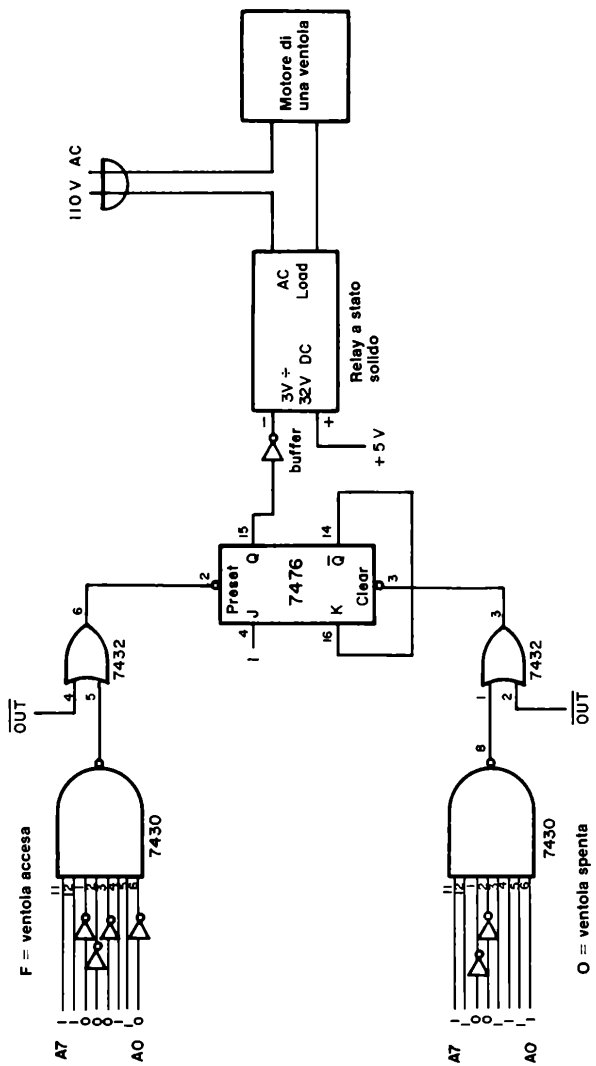
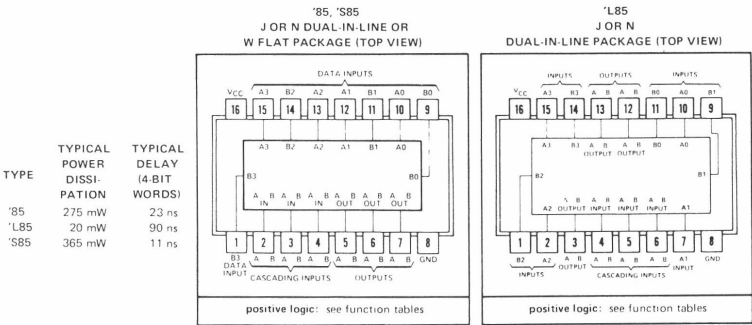



*Figura 17-8.* Circuito con relè a stato solido che impiega due parte NAND a 8 ingressi 7430 per decodificare in assoluto due impulsi di selezione dispositivo di uscita.

Un ultimo circuito di decodifica consiste di due chip comparatori 7485, dei quali diamo qui di seguito la disposizione dei pin e la tabella della verità:

**Descrizione**

Questi comparatori a grandezza assoluta a quattro bit danno il confronto diretto di codici binari e di codici BCD (8-4-2-1) vengono prese tre decisioni completamente decodificate relativamente a due parole a 4 bit (A,B), e rese esternamente disponibili su 3 uscite. Questi dispositivi sono ampliabili fino a qualunque numero di bit senza parte esterne. Per parole più lunghe, il confronto si può attuare collegando i comparatori in cascata. Le uscite A>B, A<B., A = B di uno stadio che tratta i bit meno significativi sono collegate agli ingressi corrispondenti A>B, A<B, A = B dello stadio successivo che tratta i bit più significativi. Gli stadi che trattano i bit meno significativi di tutti devono avere una tensione alto livello applicata all'ingresso A=B e, per il tipo 'L85, anche tensioni basso livello applicate agli ingressi A>B e A<B. Le linee in cascata del '85 e del 'S85 sono implementate solo con un ritardo di livello di due porte per ridurre i tempi di confronto per le parole lunghe.

FUNCTION TABLES

| COMPARING INPUTS | | | | CASCADING INPUTS | | | OUTPUTS | | |
|---|---|---|---|---|---|---|---|---|---|
| A3, B3 | A2, B2 | A1, B1 | A0, B0 | A > B | A < B | A = B | A > B | A < B | A = B |
| A3 > B3 | X | X | X | X | X | X | H | L | L |
| A3 < B3 | X | X | X | X | X | X | L | H | L |
| A3 = B3 | A2 > B2 | X | X | X | X | X | H | L | L |
| A3 = B3 | A2 < B2 | X | X | X | X | X | L | H | L |
| A3 = B2 | A2 = B2 | A1 > B1 | X | X | X | X | H | L | L |
| A3 = B3 | A2 = B2 | A1 < B1 | X | X | X | X | L | H | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 > B0 | X | X | X | H | L | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 < B0 | X | X | X | L | H | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | H | L | L | H | L | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | L | H | L | L | H | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | L | L | H | L | L | H |
| '85, 'S85 | | | | | | | | | |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | X | X | H | L | L | H |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | H | H | L | L | L | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | L | L | L | H | H | L |
| 'L85 | | | | | | | | | |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | L | H | H | L | H | H |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | H | L | H | H | L | H |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | H | H | H | H | H | H |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | H | H | L | H | H | L |
| A3 = B3 | A2 = B2 | A1 = B1 | A0 = B0 | L | L | L | L | L | L |

H = high level, L = low level, X = irrelevant

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 5012 • DALLAS, TEXAS 75222

Vi diamo informazioni sia per i chip a grande capacità (7485, 74S85) che per quelli a piccola capacità (74L85), dato che presentano una diversa configurazione dei pin ed una diversa tabella della verità. Se volete minimizzare il fan-in potete usare il chip 74L85.

Come potete vedere nella Figura 17-9, la sola condizione usata è A = B. Se il byte A del bus d'indirizzo è uguale al byte B che settate agli ingressi B del 7485, otterrete un livello logico 1 all'uscita A=B del chip 7485 in alto a destra. Per ottenere l'impulso di selezione dispositivo desiderato, invertite questo segnale e poi collegatelo col segnale di controllo $\overline{\text{OUT}}$.

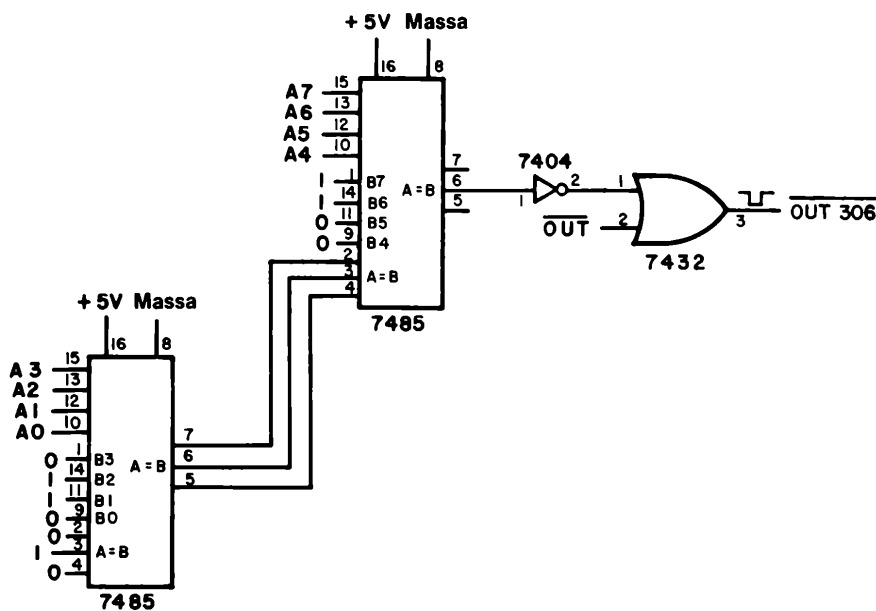



*Figura 17-9.* Circuito di decodifica che prevede l'uso di un paio di chip comparatori a 4 bit 7485. Questo circuito produce un singolo impulso di selezione dispositivo per la decodifica in assoluto. Comunque, potete cambiare il codice dispositivo alterando semplicemente l'ingresso B a 8 bit dei comparatori. In questo caso, l'ingresso B corrisponde a 11000110, 306 in codice ottale.

## INSTRUZIONI DI INGRESSO/USCITA

Vi sono soltanto due istruzioni di ingresso/uscita nell'8080A:

323 <B2>OUT — Poni il codice di dispositivo a 8 bit sul bus d'indirizzo, i contenuti dell'accumulatore sul bus di dati bidirezionali, e genera un segnale di controllo $\overline{\text{OUT}}$. I contenuti dell'accumulatore rimangono invariati.

333 <B2>IN — Poni il codice di dispositivo a 8 bit sul bus d'indirizzo, inserisci i dati presenti sul bus dati bidirezionali nell'accumulatore, e genera un segnale di controllo $\overline{\text{IN}}$.

Il secondo byte di ogni istruzione è il codice dispositivo a 8 bit. Usate il segnale di controllo e l'informazione sul bus d'indirizzo per generare l'impulso di selezione dispositivo richiesto. Un modo più conciso di fornire le istruzioni suddette è il seguente:

323 <B2>OUT — Poni in uscita i contenuti dell'accumulatore verso il dispositivo d'ingresso selezionato dal codice dispositivo nel secondo byte.

333 <B2>IN — Poni nell'accumulatore i dati presentati dal dispositivo selezionato dal codice dispositivo nel secondo byte.

Benchè gli impulsi di selezione dispositivo vengano usati di frequente per il trasferimento delle informazioni fra l'accumulatore e un dispositivo di ingresso/uscita, essi vengono impiegati anche per abilitare una operazione dei dispositivi di ingresso/uscita nei casi in cui il trasferimento dei dati non sia da o verso l'accumulatore.

## I CICLI MACCHINA FETCH, INPUT E OUTPUT

Poichè vi abbiamo descritto parecchi circuiti che possono essere usati per generare impulsi di selezione dispositivo, vi forniremo ora brevemente esempi di programmazione che illustrano l'andamento delle istruzioni IN e OUT, che sono entrambe istruzioni a due byte. Se eseguite l'istruzione in IN o OUT e controllate i contenuti del bus dati a 8 bit, osserverete qualcosa di insolito: *appare un terzo byte che non corrisponde ad un byte presente in quel punto nel vostro programma.* Che cos'è questo byte in più? È il byte che è stato trasferito da o verso il registro accumulatore 8080A durante un ciclo di istruzione IN o OUT. È durante l'esecuzione di questo terzo ciclo macchina che:

- Sul bus di controllo viene generato o un impulso $\overline{IN}$ o un impulso $\overline{OUT}$
- Il codice dispositivo appare sul bus d'indirizzo a 16 bit come due byte identici di 8 bit
- Il bus dati bidirezionale esterno e il bus dati interno, all'interno del microprocessor, permettono la comunicazione dati diretta fra l'accumulatore e il dispositivo di ingresso o di uscita.

Quando avanzate di un passo (single step) con il programma di un microcomputer 8080A, lo fate con i *cicli macchina,* e non con i byte di istruzione. Senza scendere molto in particolari, possiamo definire il ciclo macchina nel modo seguente:

*Ciclo macchina* — Una suddivisione di un ciclo di istruzione, contemporaneamente al quale, all'interno del chip del microprocessor, ha luogo un insieme di azioni ad esso collegate. Tutte le istruzioni sono combinazioni di uno o più cicli macchina.

Un esempio di ciclo macchina, è il *ciclo macchina FETCH,* durante il quale il codice di istruzioni viene prelevato dalla posizione di memoria indirizzata dal Program Counter. Durante questo ciclo, si svolgono anche semplici operazioni aritmetiche e logiche che coinvolgono il registro interno dell'8080A.

L'istruzione di uscita, OUT, consiste di due cicli macchina di FETCH sequenziali, cioè il codice istruzione e poi il codice dispositivo, seguiti da un ciclo macchina di uscita - il terzo passo che osservate eseguendo un'istruzione OUT - durante il quale potete accedere ai contenuti dell'accumulatore sul bus dati bidirezionali. Il codice dispositivo d'uscita appare sul bus d'indirizzo come due byte identici, e viene generato un impulso $\overline{OUT}$. L'istruzione IN consiste di due cicli macchina di FETCH sequenziali, cioè il codice di istruzione e poi il codice dispositivo, seguiti dal ciclo macchina di ingresso - il terzo passo che osservate eseguendo un'istruzione IN - durante il quale il buffer/latch d'ingresso, all'interno del chip 8080A, è abilitato all'ingresso dei dati sul bus dati bidirezionali: tali dati vanno poi trasferiti all'accumulatore. Sul bus d'indirizzo, appaiono due byte che rappresentano il codice dispositivo a 8 bit, e viene generato un impulso $\overline{IN}$.

## PRIMO PROGRAMMA

Consideriamo dapprima il programma dato nell'esperimento No. 5 nel capitolo VI del Bugbook V

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 074 | INR A | Incrementa il contenuto dell'accumulatore di 1 |
| 001 | 323 | OUT | Metti in uscita il contenuto dell'accumulatore sul dispositivo indicato nel byte successivo |
| 002 | 002 | 002 | Codice di dispositivo della porta 2 |
| 003 | 303 | JMP | Salto incondizionato all'indirizzo di memoria dato dai due byte seguenti |
| 004 | 000 | – | Byte di indirizzo LO |
| 005 | 003 | – | Byte di indirizzo HI |

Se eseguiste questo programma usando un circuito sigle step sul bus dati bidirezionale osservereste, in successione, i seguenti byte:

| Byte sul bus d'indirizzo | Byte sul bus dati | Commento |
|---|---|---|
| 000 | 074 | Ciclo macchina di FETCH per il codice d'istruzione INR A |
| 001 | 323 | Ciclo macchina di FETCH per il codice di istruzione OUT |
| 002 | 002 | Ciclo macchina di FETCH per il codice di dispositivo per la porta 2 |
| 002* | *contenuto dell'accumulatore* | Ciclo macchina di uscita, durante il quale il contenuto dell'accumulatore è accessibile sul bus di dati bidirezionale e sul bus d'indirizzo appare il codice di selezione dispositivo. Durante questo ciclo macchina viene generato anche un impulso $\overline{OUT}$ |
| 003 | 303 | Ciclo macchina di FETCH per il codice d'istruzione JMP |
| 004 | 000 | Ciclo macchina di FETCH per il bye d'indirizzo LO |
| 005 | 003 | Ciclo macchina di FETCH per il byte d'indirizzo HI |

* NOTA: Questo è l'indirizzo dispositivo di ingresso/uscita che appare tra i bit A0 a A7 sul bus d'indirizzo.

Osservate tale informazione sul bus di dati (a) perché tutti i byte d'istruzione si spostano sul bus di dati dalla memoria al registro istruzioni all'interno del chip 8080A, e (b) il contenuto dell'accumulatore è messo in uscita sul bus dati durante il terzo ciclo macchina dell'istruzione OUT.

Il programma incrementa il contenuto dell'accumulatore nel corso di ogni loop. Inoltre, mette in uscita il contenuto sulla porta 2 ogni volta che passa attraverso il loop. Potete osservare ciò in una porta a 8 bit che incrementa da $00000000_2$ a $11111111_2$ e poi ripete la sequenza di conteggio.

## SECONDO PROGRAMMA

Nel secondo programma, mostrato di seguito, modificate il codice dispositivo nell'istruzione OUT. In questo programma, incontrerete per la prima volta l'utilizzo delle istruzioni relative alla coppia di registri, LXI H e l'uso della coppia di registri per definire indirizzi a 16 bit di locazioni di memoria M, ed ancora l'uso di una istruzione con riferimento in memoria, INR M. Da questo punto in poi, se incontrerete delle istruzioni a voi non note, fate riferimento all'elenco delle istruzioni fornite nel capitolo 18.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 314 | 074 | INR A | Incrementa il contenuto dell'accumulatore di 1 |
| 315 | 323 | OUT | Metti in uscita il contenuto dell'accumulatore sul dispositivo dato dal codice di dispositivo memorizzato nella posizione di memoria HI=003 e LO=316 |
| 316 | <B2> | <B2> | Codice di selezione per il dispositivo di uscita |
| 317 | 041 | LXI H | Carica in modo immediato due byte nella coppia di registri H |
| 320 | 316 | <B2> | Byte di registro L |
| 321 | 003 | <B3> | Byte di registro H |
| 322 | 064 | INR M | Incrementa il contenuto della posizione di memoria indicata dalla coppia di registri H |
| 323 | 303 | JMP | Salto incondizionato all'indirizzo di memoria dato nei due byte seguenti |
| 324 | 314 | – | Byte di indirizzo LO |
| 325 | 003 | – | Byte di indirizzo HI |

Questo programma vi permette di mettere in uscita il contenuto dell'accumulatore su 256 diversi dispositivi in sequenza, iniziando su quello dato a HI = 003 e LO = 316. Ad ogni loop del programma, il codice di dispositivo a LO = 316 è incrementatato di uno. Questo non è un programma molto utile, ma sta a dimostrare il fatto che il codice di dispositivo non rimane inviolato all'interno di un programma. Con pochissime istruzioni, potete cambiare il codice di dispositivo e passare così attraverso una serie di dispositivi successivi l'uno dall'altro. In pratica, il codice di selezione del dispositivo di uscita che vi interessa sarebbe probabilmente memorizzato in un registro, e si userebbe un'istruzione MOV per trasferire il contenuto del registro alla posizione di memoria M indirizzata dalla coppia di registri H.

Otterreste un risultato utile se questo programma fosse in ROM, PROM o EPROM?
No, perché in tal caso non sareste in grado di cambiare il contenuto della posizione di memoria LO = 316.

## INTRODUZIONE AGLI ESPERIMENTI

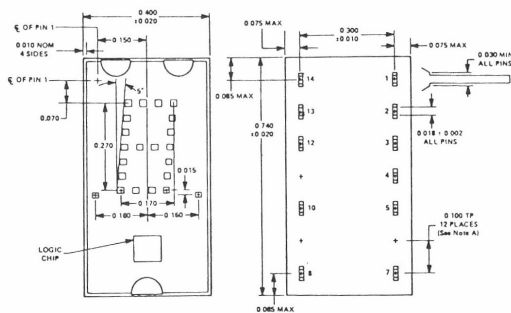
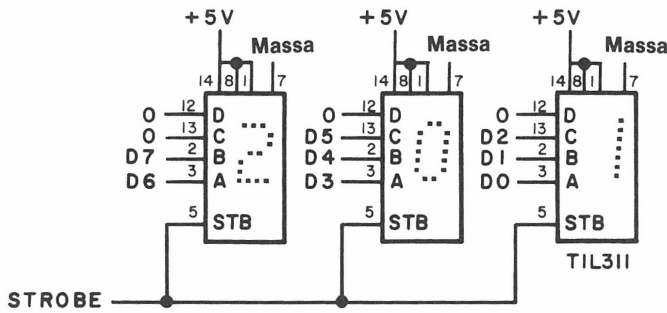
Gli esperimenti seguenti dimostrano come potete generare, ed usare, impulsi di selezione dispositivo. Inoltre cablerete un "bus monitor" e diverrete esperti nell'uso di un circuito monostabile.

| Esperimento | Commento |
|---|---|
| 1 | Illustra il circuito di un bus monitor basato sull'indicatore numerico TIL311 che vi permette di controllare tutti i dati che passano sul bus dati bidirezionale. |
| 2 | Illustra il circuito di un bus monitor basato sull'indicatore numerico HP 5082-7300 che vi permette di controllare tutti i dati che passano sul bus dati bidirezionale. |
| 3 | Dimostra l'uso di un circuito monostabile per il microcomputer MMD-1. Avanzerete di un passo tramite i primi trentotto cicli macchina del programma KEX. |
| 4 | Contate gli impulsi di strobe $\overline{IN}$ e $\overline{OUT}$ con l'aiuto di un contatore 7490 mentre il microcomputer opera avanzando di un solo passo. Determinate inoltre la configurazione dei bit per la tastiera. |
| 5 | Costruite, attivate e provate un circuito di interfaccia che vi permetterà di generare sedici diversi impulsi di selezione dispositivo. |
| 6 | Dimostra come i canali decodificati sulla piastra del circuito stampato MMD-1 possono essere usati per generare impulsi di selezione dispositivo. |
| 7 | Dimostra l'uso di un impulso di selezione dispositivo per azzerare un contatore 7490. |
| 8 | Dimostra come si usano due chip comparatori 7485 per generare un impulso di selezione dispositivo decodificato in assoluto. |
| 9 | Dimostra come si usa una porta NAND a 8 ingressi 7430 per generare un impulso di selezione dispositivo decodificato in assoluto. Dimostra anche come si usano due di tali impulsi, un flip-flop 7476, e un relè a stato solido per accendere e spegnere il motore di un ventilatore. |

## ESPERIMENTO N. 1

### Scopo

L'esperimento consiste nel cablare un *bus monitor,* un display ottale a tre cifre che controlla i dati che appaiono sul bus di dati bidirezionale.

### Configurazione dei pin dell'indicatore numerico

### Schema del circuito

### Passo 1

Montate il circuito mostrato nella figura oppure usate l'LR-27 bus monitor Outboard,. Quando montate il circuito suddetto, *nonché qualunque altro circuito di interfaccia in questo Bugbook,* fatelo dopo aver tolto l'alimentazione al microcomputer!

### Passo 2

Collegate l'ingresso STROBE del latch al livello logico 0.
Vengono così abilitati i display, in modo che tutti i display dell'indicatore numerico seguono gli ingressi.

Alimentate microcomputer MMD-1. Dovreste osservare che la maggior parte dei punti nell'indicatore numerico sono illuminati. Ciò che state vedendo è il loop di attesa nella Keyboard EXsecutive EPROM, che viene eseguito ad un clock rate di 750 kHz. L'unica cosa che potete apprendere dal fatto che tutti gli indicatori numerici sono illuminati, è che il microcomputer sta eseguendo un programma.

**Passo 3**

Ora collegate l'ingresso STROBE alla linea del segnale di controllo $\overline{OUT}$ sul foro di connessione del bus SK-10. Se non riuscite a trovare $\overline{OUT}$, fate riferimento alla Figura 16-3.

**Passo 4**

Premete il tasto RESET sul microcomputer MMD-1. Quale byte a tre cifre ottali appare sul bus monitor? Scrivetelo qui sotto.

Quale byte a tre cifre ottali appare alla Porta 2? Scrivetelo qui sotto.

Sono gli stessi?

Sì.

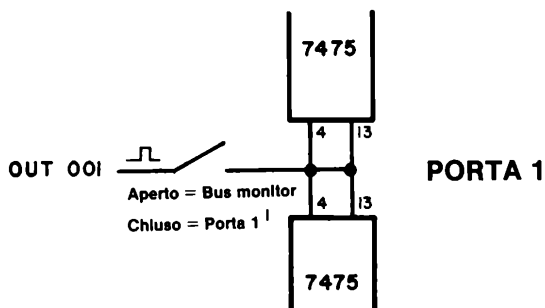
**Passo 5**

Qual'è il significato dell'informazione che appare sul bus monitor quando l'ingresso STB è collegato con $\overline{OUT}$? Se non siete sicuri di saper rispondere a questa domanda, caricate dei valori ottali arbitrari nella memoria di lettura/scrittura ed osservate quale informazione appare sul bus monitor. Esaminate anche il vostro "programma" ed osservate di nuovo la relazione esistente fra il byte sul bus monitor ed il byte visualizzato alla Porta 2. Che cosa concludete?

Concludiamo che l'informazione sul bus monitor (una porta di uscita per il microcomputer) e quella alla Porta 2 sono identiche. Le due porte di uscita danno il contenuto della posizione di memoria indirizzata dall'indirizzo di 16 bit dato nelle Porte 0 e 1, oppure dal byte che va caricato in memoria, o dal registro indirizzi LO, o dal registro HI. Il bus monitor rende più facile l'inserimento e il controllo di un programma.

*Conservate questo circuito per tutti gli altri esperimenti di questo Bugbook. L'esperimento n⁰ 2 è simile a questo, ma usa un indicatore numerico diverso, l'HP 5082-7300.*

**Appendice all'esperimento N. 1**

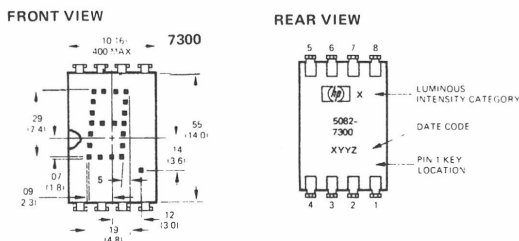
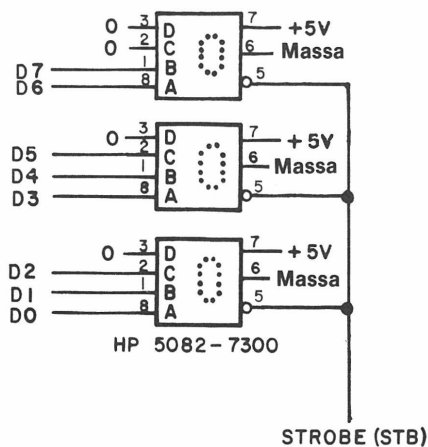
Sulla Porta 1 del microcomputer MMD-1, si può costruire un circuito meno costoso. Invece che da tre cifre ottali, il monitor è costituito da otto LED che controllano continuamente lo stato del bus dati bidirezionale, da D0 a D7. Per costruire questo bus monitor, mettete un interruttore sulla linea d'ingresso ENABLE dei chip 7475 della Porta 1, come mostra la figura:

Quando l'interruttore è chiuso, i chip 7475 IC24 e IC25 operano normalmente come Porta 1. Quando l'interruttore è aperto, essi operano come un bus monitor a 8 bit.

Esattamente in questo modo, questa modifica viene inserita nei futuri modelli del microcomputer MMD-1.

## ESPERIMENTO N. 2

### Scopo

Questo esperimento consiste nel cablare un *bus monitor* usando l'indicatore numerico HP 5082-7300. Questo esperimento è identico all'esperimento n⁰ 1.

### Configurazione dei pin dell'indicatore numerico

### Schema del circuito

### Passo 1

Cablate il circuito mostrato nella figura. Vi raccomandiamo di farlo dopo aver tolto la corrente al microcomputer.

### Passo 2

Collegate l'ingresso STROBE del latch al livello logico 0. Si abilitano così i display, in modo che tutti i display dell'indicatore numerico seguono gli ingressi.

Alimentate il microcomputer MMD-1. Dovreste osservare che la maggior parte dei punti dell'indicatore numerico sono illuminati. Ciò che state vedendo è il loop di attesa della Keyboard EXecutive EPROM, che viene eseguito ad un clock rate di 750 kHz. L'unica cosa che potete apprendere dal fatto che tutti gli indicatori numerici sono accesi, è che il microcomputer sta eseguendo un programma.

**Passo 3**

Ora collegate l'ingresso STROBE alla linea del segnale di controllo $\overline{OUT}$ sul foro di connessione del bus SK-10. Se non riuscite a trovare $\overline{OUT}$, fate riferimento alla Figura 16-3.

**Passo 4**

Premete il tasto RESET sul microcomputer MMD-1. Quale byte a tre cifre ottali appare sul bus monitor? Scrivetelo qui sotto.

Quale byte a tre cifre ottali appare alla Porta 2? Scrivetelo qui sotto.

Sono gli stessi?

Sì.

**Passo 5**

Qual'è il significato dell'informazione che appare sul bus monitor quando l'ingresso STB è collegato con $\overline{OUT}$? Se non siete sicuri di saper rispondere a questa domanda, caricate dei valori ottali arbitrari nella memoria di lettura/scrittura ed osservate quale informazione appare sul bus monitor. Esaminate anche il vostro "programma" ed osservate di nuovo la relazione esistente fra il byte sul bus monitor ed il byte visualizzato alla Porta 2. Che cosa concludete?

Concludiamo che l'informazione sul bus monitor (una porta di uscita per il microcomputer) e quella alla Porta 2 sono identiche. Le due porte di uscita danno il contenuto della posizione di memoria indirizzata dall'indirizzo di 16 bit dato nelle Porte 0 e 1, oppure dal byte che va caricato in memoria, o dal registro indirizzi LO, o dal registro HI. Il bus monitor rende più facile l'inserimento e il controllo di un programma.

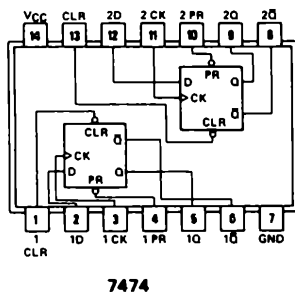
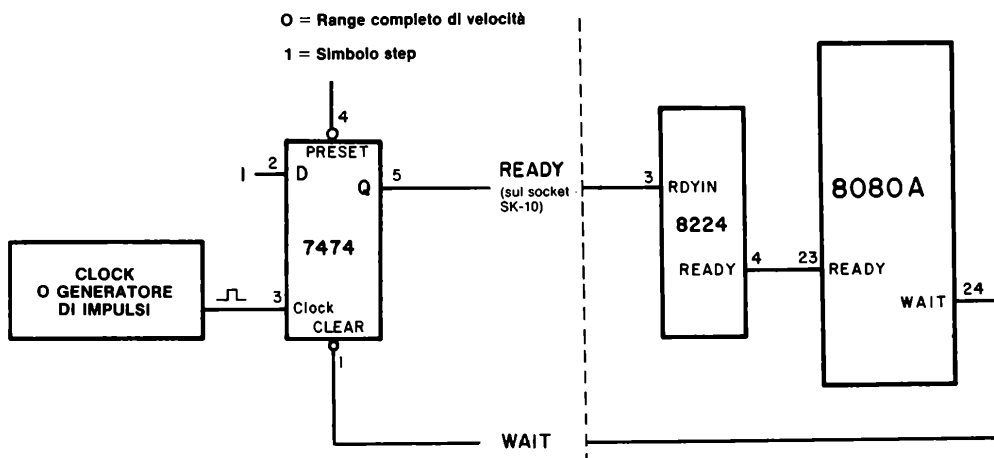
*Conservate o questo circuito o il circuito dell'esperimento n$^{0}$ 1 per tutti gli altri esperimenti di questo Bugbook.*

## ESPERIMENTO N. 3

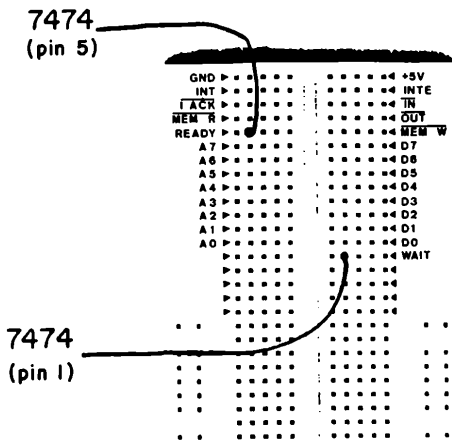
### Scopo

Questo esperimento consente di costruire un circuito monostabile per il microcomputer MMD-1

### Configurazione dei pin del circuito integrato

7474

### Schema del circuito

### Passo 1

Montate il circuito mostrato nella figura. Usate le posizioni READY e WAIT sull'SK-10. Da notare nella figura che i chip situati alla destra della linea tratteggiata sono già cablati sulla piastra a circuito stampato.

La figura seguente mostra i collegamenti specifici che dovete effettuare sul bus dell'SK-10:

**Passo 2**

Collegate l'ingresso di abilitazione del latch (STB) del bus monitor al livello logico 0 (GND-Massa). Questo vi permette di osservare tutte le informazioni che appaiono sul bus dati bidirezionale. Posizionate il circuito monostabile per l'operazione passo-passo (pin 4 del chip 7474 collegato al livello logico 1). Se state usando la piastra del monostabile, mettete lo switch logico nella posizione che corrisponde all'operazione passo-passo.

Premete il tasto RESET. Dovreste osservare un 303 sul bus monitor.
*Questo è il primo byte di istruzioni della routine KEX.*

**Passo 3**

Usando il generatore di impulsi, avanzate passo-passo con la KEX, che inizia alla posizione di memoria HI = 000 e LO = 000. Confrontate le vostre osservazioni con la sequenza di byte che osservate sul bus monitor, e che vi mostriamo di seguito. Lo scopo di questo elenco è mostrarvi come si lavora in single step. Può darsi che non capiate tutte le istruzioni seguenti.

| Indirizzo di memoria | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 000 | 303 | JMP | Salto incondizionato all'indirizzo di memoria START dato dai due byte seguenti |
| 000 001 | 070 | START | Byte di indirizzo LO di START |
| 000 002 | 000 | – | Byte di indirizzo HI di START |
| 000 070 | 061 | LXI SP | Carica in modo immediato due byte nel registro stack pointer |
| 000 071 | 000 | 000 | Byte dello stack pointer LO |

| | | | |
|---|---|---|---|
| 000 072 | 004 | 004 | Byte dello stack pointer HI |
| 000 073 | 041 | LXI H | Carica in modo immediato due byte nei due registri H |
| 000 074 | 000 | 000 | Byte del registro L |
| 000 075 | 003 | 003 | Byte del registro H |
| 000 076 | 116 | MOV C, M | Poni il contenuto della posizione di memoria M (che è indicata dai due registri H) nel registro C |
| *003 000* | *XYZ* | *XYZ* | *Ciclo macchina MEMORY READ, nel quale i contenuti della posizione di memoria HI = 003 e LO = 000 sono spostati nel registro C. Osservate questo byte di memoria sul bus monitor. Il vostro valore XYZ è il valore contenuto nella prima posizione di memoria lettura/scrittura del vostro microcomputer MMD-1.* |
| 000 077 | 174 | MOV A, H | Poni il contenuto del registro H nell'accumulatore |
| 000 100 | 323 | OUT | Poni in uscita il contenuto dell'accumulatore sulla porta di uscita data nel byte seguente |
| 000 101 | 001 | 001 | Codice di dispositivo per la porta di uscita 1 |
| *001 001* | *003* | *003* | *Ciclo macchina di OUTPUT, durante il quale il contenuto dell'accumulatore è messo in uscita sulla porta 1. Il codice di dispositivo è messo in uscita come due byte 001 identici sul bus d'indirizzo per il chip 8080A.* |
| 000 102 | 175 | MOV A, L | Poni il contenuto del registro L nell'accumulatore. |
| 000 103 | 323 | OUT | Poni in uscita il contenuto dell'accumulatore sulla porta di uscita data nel byte seguente |
| 000 104 | 000 | 000 | Codice di dispositivo per la porta di uscita 0 |
| *000 000* | *000* | *000* | *Ciclo macchina di OUTPUT, durante il quale il contenuto dell'accumulatore viene messo in uscita sulla Porta 0. Il codice di dispositivo è messo in uscita come due byte 000 identici sul bus d'indirizzo.* |
| 000 105 | 171 | MOV A, C | Poni il contenuto del registro C nell'accumulatore |

| | | | |
|---|---|---|---|
| 000 106 | 323 | OUT | Poni in uscita il contenuto dell'accumulatore sulla porta di uscita data nel byte seguente |
| 000 107 | 002 | 002 | Codice di dispositivo per la porta di uscita 2 |
| *002 002* | *XYZ* | *XYZ* | *Ciclo macchina di OUTPUT, durante il quale il contenuto dell'accumulatore viene messo in uscita sulla porta 2. Questo è il byte estratto in precedenza dalla memoria di lettura/scrittura. Il codice di dispositivo è messo in uscita sul bus d'indirizzo come due byte 002 identici.* |
| 000 110 | 315 | CALL | Subroutine di richiamo KBRD posizionata all'indirizzo di memoria dato dai due byte d'indirizzo seguenti |
| 000 111 | 315 | KBRD | Byte d'indirizzo LO di KBRD |
| 000 112 | 000 | – | Byte d'indirizzo HI di KBRD |
| *003 377* | *000* | *000* | *Ciclo macchina STACK WRITE, durante il quale il byte d'indirizzo HI nel contatore di programma viene spostato nello stack in memoria* |
| *003 376* | *113* | *113* | *Ciclo macchina STACK WRITE, durante il quale il byte d'indirizzo LO nel contatore di programma viene spostato nello stack in memoria* |
| 000 315 | 333 | IN | Byte d'ingresso nell'accumulatore dalla porta d'ingresso data dal codice di dispositivo seguente |
| 000 316 | 000 | 000 | Codice di dispositivo per la tastiera sul microcomputer MMD-1 |
| *000 000* | *160* | *160* | *Ciclo macchina di INPUT, durante il quale viene inserito un byte dalla tastiera. Il byte, 160, viene inserito se non premete nessun tasto. Il codice di dispositivo viene messo in uscita sul bus d'indirizzo come due byte 000 identici.* |
| 000 317 | 267 | ORA A | OR del contenuto dell'accumulatore con se stesso |
| 000 320 | 372 | JMP | Se il contenuto dell'accumulatore è negativo (il bit D7 a livello logico 1) torna indietro alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 000 321 | 315 | .315 | Byte d'indirizzo LO |

| | | | |
|---|---|---|---|
| 000 322 | 000 | 000 | Byte d'indirizzo HI |
| 000 323 | 315 | CALL | Richiama la subroutine TIMOUT posizionata all'indirizzo di memoria dato dai due byte d'indirizzo seguenti |
| 000 324 | 277 | TIMOUT | Byte d'indirizzo LO di TIMOUT, una subroutine a ritardo di tempo di 10 ms |
| 000 325 | 000 | – | Byte d'indirizzo HI di TIMOUT |
| . | . | | |
| . | . | | |
| . | . | | |

Ci fermeremo qui, alla subroutine di delay. Chiaramente, il valore di questi tipi di circuiti (circuito monostabile e circuito di un bus monitor) sta nel fatto che vi permettono di osservare i dati che vengono trasferiti fra la memoria, o gli organi di ingresso/uscita, e l'interno del chip del microprocessor 8080A, cioè siete in grado di osservare cicli macchina anzichè cicli di FETCH. Questo è particolarmente importante quando controllate e provate nuovi programmi.

**Passo 4**

Riportare il computer alla massima velocità operativa, 750 kHz.

Ora caricate 377 nel registro HI (porta 1), nel registro LO (porta O) e nel registro dati (porta 2). Ora tutti e ventiquattro gli indicatori a LED dovrebbero essere accesi. Eseguite KEX avanzando di un passo e incominciando ancora una volta da HI = 000 e LO = 000. Quando cambia da 377 a 003 il registro HI? Dovrete far avanzare il programma di un passo almeno fino alla prima istruzione OUT. Perchè?

Al ciclo macchina che segue immediatamente l'esecuzione del byte di istruzioni a HI = 000 e LO = 101. Questi dati cambiano soltanto durante l'esecuzione di un'istruzione OUT 001. Questi LED non sono collegati al bus d'indirizzo. Sono usati solo per rappresentare il byte d'indirizzo HI.

**Passo 5**

Continuate ad avanzare di un passo attraverso KEX. Quando cambia da 377 a 000 il registro LO?

Al ciclo macchina che segue immediatamente l'esecuzione del byte di istruzioni a HI = 000 e LO = 104. Questi dati cambiano soltanto durante l'esecuzione di un'istruzione OUT 000.

**Passo 6**

Continuate ad avanzare di un passo con KEX. Quando cambia il registro dati da 377 ad un qualunque valore memorizzato nella posizione di memoria HI = 003 e LO = 000?

Al ciclo macchina che segue immediatamente l'esecuzione del byte di istruzioni a LO = 107. Questi dati cambiano soltanto durante l'esecuzione di un'istruzione OUT 002.

*Conservate il vostro bus monitor e i circuiti e passate al prossimo esperimento.*

## ESPERIMENTO N. 4

### Scopo

Questo esperimento permette di conteggiare gli impulsi di ingresso e uscita, $\overline{IN}$ e $\overline{OUT}$, con l'aiuto di un contatore 7490 mentre il microcomputer viene fatto avanzare di un passo.

### Configurazione dei pin del circuito integrato

### Schema del circuito

### Programma

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | IN | Byte in ingresso dalla tastiera verso l'accumulatore |
| 001 | 000 | 000 | Codice di dispositivo per la tastiera |
| 002 | 323 | OUT | Contenuto in uscita dell'accumulatore sulla porta di uscita nel byte seguente |

| | | | |
|---|---|---|---|
| 003 | 000 | 000 | Codice di dispositivo per la porta 0 |
| 004 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 005 | 000 | – | Byte d'indirizzo LO |
| 006 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Questo è un programma interessante, poiché dimostra una serie di importanti concetti associati con le istruzioni di ingresso e di uscita e con le operazioni del microcomputer MMD-1.

**Passo 2**

Prima di montare il circuito 7490, caricate il programma suddetto in memoria ed eseguitelo a 750 kHz. Quale configurazione di bit osservate alla porta 0?

Osserviamo $01110000_2$ alla porta 0.

**Passo 3**

Adesso premete i seguenti tasti uno dopo l'altro: 0, 1, 2, 3, 4, 5, 6, e 7. Nello spazio qui sotto scrivete la configurazione di bit che osservate alla porta 0 per ognuno di questi tasti. Che correlazione esiste fra il numero dei tasti e la configurazione dei bit?

Abbiamo osservato quanto segue:

| Tasto | Configurazione dei bit alla porta 0 |
|---|---|
| 0 | 11110000 |
| 1 | 11110001 |
| 2 | 11110010 |
| 3 | 11110011 |
| 4 | 11110100 |
| 5 | 11110101 |
| 6 | 11110110 |
| 7 | 11110111 |

Notate che, in qualunque momento premiate un tasto, il bit D7 va al livello logico 1. Per i tasti da 0 a 7, i tre bit meno significativi corrispondono agli equivalenti ottali del tasto.

**Passo 4**

Premete i tasti rimanenti ad eccezione di RESET. Scrivete qui sotto la configurazione di bit che osservate.

Osserviamo quanto segue:

| Tasto | Configurazione di bit alla porta 0 |
|---|---|
| S | 11111000 |
| C | 11111010 |
| G | 11111011 |
| H | 11111100 |
| L | 11111101 |
| A | 11111110 |
| B | 11111111 |

In qualunque momento premiate un tasto, di nuovo il bit D7 va a livello logico 1. Questo è il bit usato da KEX per determinare se un tasto viene premuto oppure no. Fate riferimento all'esperimento Nº 2 di *questo capitolo* e all'istruzione LO = 320. È in questo caso che KEX rivela il fatto che un tasto venga premuto.

**Passo 5**

Montate il circuito del contatore e collegate l'uscita $\overline{OUT}$ del contatore. Mentre il microcomputer esegue il programma a piena cadenza di temporizzazione, commutate lo switch logico (o il filo) collegato ai 4 ingressi del pin 7474 sullo stato di livello logico 1. Il microcomputer opera ora a passo - passo.

**Passo 6**

Se non avete un bus monitor, tutto quello che sarete in grado di fare in questo esperimento, è contare gli impulsi del segnale di controllo $\overline{OUT}$. Avanzate di un passo con il programma e osservate il risultato: ottenete un solo conteggio per ogni nove volte che il generatore di impulsi viene premuto e lasciato andare. Se l'uso del generatore diventa noioso, sostituitelo con un clock Outboard che opera ad un frequenza di circa 0,3 ÷ 1 kHz.

**Passo 7**

Staccate il filo che collega $\overline{OUT}$ al pin 14 del contatore 7490. Collegate $\overline{IN}$ al pin 14. $\overline{IN}$ è adiacente ad $\overline{OUT}$ sul socket di breaboarding SK-10. Continuate ad eseguire il programma in single-stop. Dovreste osservare un solo conteggio sul contatore 7490 per ogni nove impulsi di clock applicati al circuito monostabile.

**Passo 8**

Se avete un bus monitor, collegate l'ingresso di abilitazione del latch (STB) al livello logico 0. Ora avanzate di un passo con il programma. Dovreste osservare la sequenza di byte che segue. Studiate tale sequenza anche se non avete un bus monitor:

| Indirizzo di memoria | Byte di istruzioni | Codice mnemonico | Descizione |
|---|---|---|---|
| 003 000 | 333 | IN | Inserimento del byte dalla tastiera nell'accumulatore |
| 003 001 | 000 | 000 | Codice di dispositivo per la tastiera sul microcomputer MMD-1 |
| *000 000* | *160* | *160* | *Ciclo macchina INPUT, durante il quale viene inserito un byte dalla tastiera. Il byte, 160, è inserito se non premete nessun tasto. Il codice di dispositivo è messo in uscita sul bus d'indirizzo come due byte 000 identici.* |
| 003 002 | 323 | OUT | Poni in uscita il contenuto dell'accumulatore sulla porta di uscita data nel byte seguente |
| 003 003 | 000 | 000 | Codice di dispositivo per la porta 0. |
| *000 000* | *160* | *160* | *Ciclo macchina OUTPUT, durante il quale il contenuto dell'accumulatore è messo in uscita sulla porta 0. Questo è il byte che viene inserito dalla tastiera. Il codice di dispositivo è messo in uscita sul bus d'indirizzo come due byte 000 identici.* |
| 003 004 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte seguenti |
| 003 005 | 000 | 000 | Byte d'indirizzo LO per l'avviamento del programma |
| 003 006 | 003 | 003 | Byte d'indirizzo HI per l'avviamento del programma |

.
ecc.
.

Questo programma si ripete ogni nove cicli macchina. Perché occorrono nove passi per l'esecuzione di un programma a sette byte?

Ogni passo è un "ciclo macchina", e le istruzioni IN e OUT richiedono un ciclo macchina aggiuntivo per essere eseguite. Questo ciclo macchina extra è predefinito all'interno del chip 8080A ed è caratteristico anche di altri tipi di istruzione, comprese le istruzioni di riferimento alla memoria, i richiami, i ritorni, PUSH e POP.

*In questo esperimento, il fatto che inizialmente dobbiate eseguire un programma a 750 kHz prima di iniziare ad operare avanzando di un passo, può crearvi delle difficoltà. Ma se dimenticate di farlo, non abbandonerete più il programma KEX.*

**Passo 9**

Premete il tasto 7, e tenetelo premuto. Se avete un bus monitor, quale byte appare durante i cicli macchina INPUT e OUTPUT?

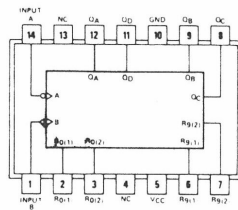
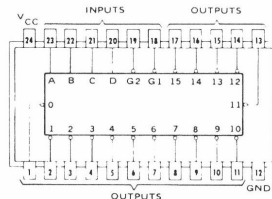
Abbiamo osservato il byte 367 nel corso di entrambi i cicli di macchina, INPUT e OUTPUT. Questo byte è stato anche messo in uscita sulla porta 0. Se siamo in grado di controllare le informazioni sul bus dati bidirezionale, siamo anche in grado di osservare lo spostamento dei dati dalla tastiera nell'accumulatore, e dall'accumulatore nel latch della porta 0.

*Conservate il contatore 7490 e i circuiti e passate al prossimo esperimento.*

## ESPERIMENTO N. 5

### Scopo

Questo esperimento permette di costruire un circuito decodificatore, basato su di un chip decodificatore 74154, che può generare sedici diversi impulsi di selezione dispositivo.

### Configurazioni dei pin dei circuiti integrati.

7490 74154

### Schema del circuito

+5V GND

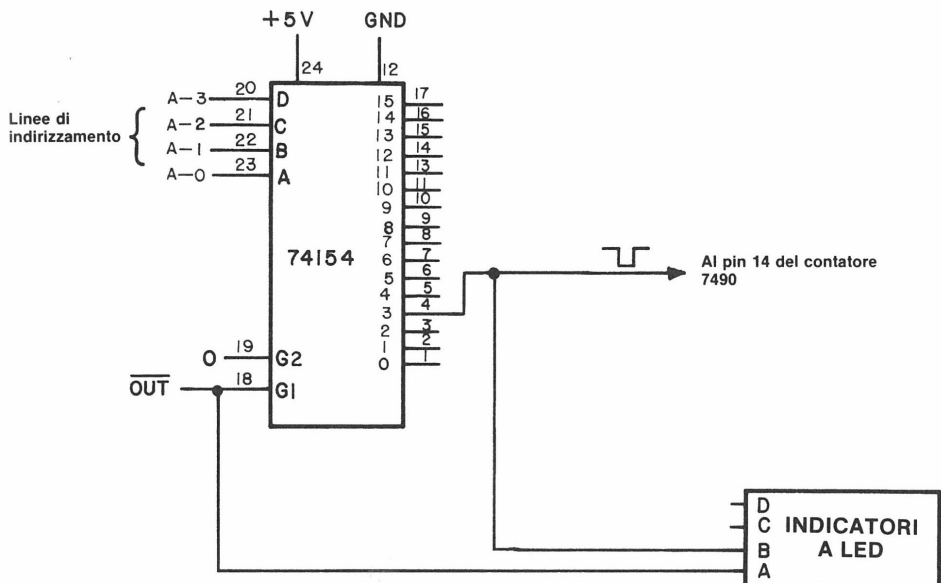
Linee di indirizzamento

A—3 A—2 A—1 A—0

74154

OUT

Al pin 14 del contatore 7490

INDICATORI A LED

**Programma**

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | IN | Generazione di un impulso di selezione per il dispositivo d'ingresso dato nel byte seguente |
| 001 | 000 | 000 | Codice di selezione per il dispositivo d'ingresso 000 |
| 002 | 323 | OUT | Generazione di un impulso di selezione dispositivo di uscita per il dispositivo di uscita dato dal byte seguente |
| 003 | <B2> | <B2> | Codice di selezione per il dispositivo di uscita |
| 004 | 303 | JMP | Salto incondizionato indietro verso l'inizio del programma, il cui indirizzo è dato dai due byte d'indirizzo seguenti |
| 005 | 000 | – | Byte d'indirizzo LO |
| 006 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Nel circuito mostrato, potete generare sedici impulsi di selezione dispositivo di uscita consecutivi. Montate il circuito usando un chip decodificatore 74154.

Se avete un bus monitor ottale a tre cifre, volendo potete osservare il contenuto del bus dati bidirezionale nell'eseguire il programma avanzando di un passo. Montate l'ingresso di abilitazione del latch (STB) sul bus monitor collegandolo al livello logico 0.

**Passo 2**

Nel programma, usate il codice di dispositivo 003 a LO = 003. Caricate il programma nella memoria di lettura/scrittura. Assicuratevi che l'indicatore a LED e il pin 14 del contatore 7490 siano collegati al pin 4 del decodificatore 74154.

**Passo 3**

Eseguite il programma avanzando di un passo. Fate il conto di quanti impulsi di segnale di controllo $\overline{\text{OUT}}$ sono necessari ogni nove cicli macchina, e scrivete la risposta nello spazio che segue.

Dovreste osservare un impulso $\overline{\text{OUT}}$ ogni nove cicli macchina.

**Passo 4**

È uso comune indicare un impulso di selezione dispositivo con $\overline{\text{DS xxx}}$, se è un impulso a livello logico 0 (o negativo), e con DS xxx se è un impulso a livello logico 1 (o positivo). Le lettere 'xxx' indicano il codice dispositivo ottale a tre cifre. La sbarretta scritta sopra un codice di impulsi funzionale è la notazione standard di un livello logico 0 attivo.

Nello schema del circuito, vi è un collegamento tra il canale di uscita N. 3 del decodificatore 74154 e il contatore 7490. Mentre il microcomputer opera alla massima velocità, provate alcune delle altre uscite del decodificatore e stabilite se vi è un altro canale che genera un impulso di uscita $\overline{\text{DS 003}}$. Che canale è?

Il canale 3 al pin 4 sul chip dovrebbe essere l'unico a far operare il contatore ad una cadenza veloce. Tutti gli altri canali non sono funzionali per un codice di dispositivo 003.

**Passo 5**

Ora cambiate il byte di istruzioni LO = 003 in LO = 017. Su quale canale di uscita del decodificatore 74154 osservate l'impulso di selezione dispositivo? Quale sarebbe il modo esatto di indicare un impulso di questo tipo, cioè: DS xxx o $\overline{\text{DS xxx}}$? Che cos'è 'xxx'?

Abbiamo osservato l'impulso di selezione dispositivo al canale $15_{10}$ (pin 17)) Il modo esatto di indicare questo impulso è $\overline{\text{DS 017}}$.

**Passo 6**

Durante il ciclo macchina nel quale viene generato un impulso di selezione dispositivo, qual'è il livello logico dell'indicatore a LED A?

L'indicatore a LED A è a livello logico 0 ogniqualvolta viene eseguita un'istruzione OUT. Un ciclo macchina è una suddivisione di un ciclo di istruzioni durante il quale avvengono una serie di azioni collegate all'interno del chip del microprocessor. *Quando avanzate di un passo con un programma, lo fate attraverso i cicli macchina, non attraverso le istruzioni.*

**Passo 7**

A quale ciclo macchina viene generato un impulso di selezione dispositivo per l'istruzione OUT, il primo, il secondo o il terzo ciclo?

Il terzo ciclo macchina, I primi due cicli macchina sono cicli di fetch, che inseriscono il codice operazione 323, e il codice dispositivo, <B2>, estraendoli dalle posizioni di memoria nelle quali sono memorizzati.

**Passo 8**

Variando il byte di istruzioni LO = 003, potete variare il canale di uscita del decodificatore 74154 sul quale appare un impulso di selezione dispositivo. Per i byte del codice dispositivo dati nella tabella seguente, su quale canale di uscita 74154 appare l'impulso di selezione dispositivo? Quando cambiate il programma, ricordate che contemporaneamente deve essere in atto l'esecuzione della routine del monitor KEX a 750 kHz.

| Byte del codice dispositivo a LO = 003 | Canale di uscita del decodificatore 74154 |
|---|---|
| 000 | 0 |
| 001 | 1 |
| 002 | 2 |
| 003 | 3 |
| 010 | |
| 017 | |
| 020 | |
| 025 | |
| 050 | |
| 377 | |

Per gli ultimi sei byte del codice dispositivo, abbiamo osservato i seguenti risultati:

| | |
|---|---|
| 010 | 8 |
| 017 | 15 |
| 020 | 0 |
| 025 | 5 |
| 377 | 15 |

Se non avete ottenuto gli stessi risultati, ripetete questo passo.

**Passo 9**

Il decodificatore 74154 ha soltanto sedici canali di uscita, e vi aspettereste in un primo tempo, che esso possa decodificare soli i primi sedici codici di dispositivo di uscita: 000, 001, 002, 003, 004, ... 015, 016, e 017. Perchè osservate impulsi di selezione dispositivo per codici maggiori di 017?

Perchè il circuito decodificatore 74154 non decodifica *in assoluto* il byte del codice dispositivo. Vengono decodificati solo i quattro bit meno significativi del codice dispostitivo. Benchè possiate generare solo sedici diversi impulsi di selezione dispositivo, ogni singolo impulso può essere generato in sedici modi diversi, usando sedici diversi codici dispositivo. Questa non è una buona prassi da seguire nella progettazione delle interfacce dei micocomputer. Dovreste sempre tentare di decodificare in assoluto sia i codici dei dispositivi d'ingresso che quelli dei dispositivi di uscita.

**Passo 10**

Come decodifichereste in assoluto sedici dei 256 possibili codici dispositivo usando un decodificatore 75154 e uno o più chip aggiuntivi?

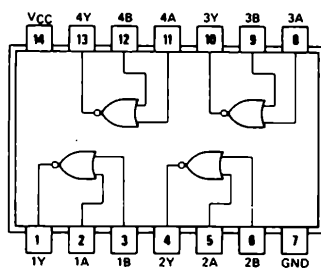
Potete usare un altro chip 74154 per abilitare il primo decodificatore 74154 al pin dell'ingresso G1. In alternativa, potete usare una porta OR a 4 ingressi per abilitare il decodificatore 74154 al pin dell'ingresso G1; i bit d'indirizzo A-4, A-5, A-6 e A-7 fungerebbero da ingressi della porta OR. Si possono usare vari schemi di decodificatori.

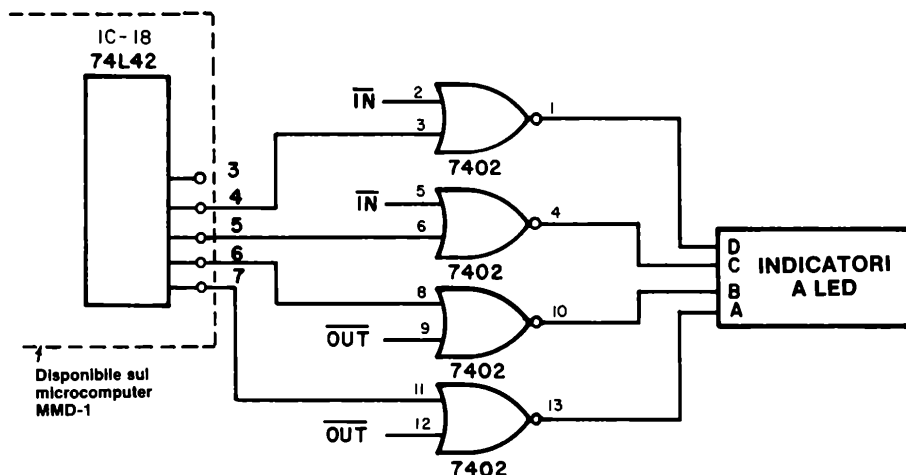
*Togliete il decodificatore 74154 e il circuito dell'indicatore a LED usati in questo esperimento. Il contatore 7490 e il circuito monostabile saranno usati in un esperimento successivo.*

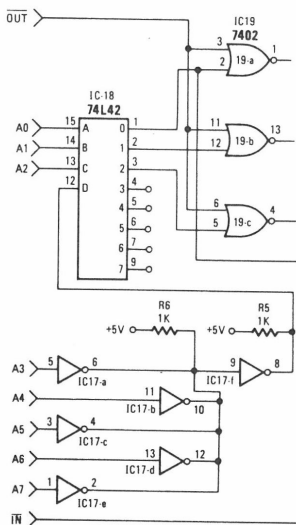
## ESPERIMENTO N. 6

### Scopo

Questo esperimento dimostra in che modo si possono usare gli indirizzi decodificati sulla piastra dell'MMD-1 per generare impulsi di selezione per dispositivi d'ingresso e di uscita.

### Configurazione dei pin del circuito integrato

7402

### Schema del circuito

Troverete cinque pin di breadboarding senza saldatura adiacenti al chip 74L42 nella sezione del decodificatore ingresso/uscita della piastra del circuito stampato.

Cercate il circuito integrato IC-18 (o A 18). La figura seguente mostra lo schema di connessione per la sezione del decodificatore ingresso/uscita:

I cerchi associati con i canali di uscita da 3 a 7 sul chip 74L42 rappresentano o le saldature o i pin di breadboarding. Notate che il calcolatore utilizza le porte NOR a 2 ingressi 7402, che generano impulsi di selezione dispositivo positivi, come DS 000, DS 001, e DS 002. Il circuito mostrato precedentemente genera impulsi di selezione per dispositivi sia d'ingresso che di uscita, che possiamo definire "IN 004", "IN 005", "OUT 006", e "OUT 007".

Notate anche che il 74L42 è un decodificatore in assoluto per il codice dispositivo a 8 bit. I bit A3, A4, A5, A6, e A7 sul bus d'indirizzo devono essere tutti a livello logico 0, affinchè l'ingresso D (pin 12) sul chip 74L42 sia a livello logico 0. I tre bit che del bus d'indirizzo rimanenti, A0, e A2 sono usati nella decodificazione di otto canali di ingresso/uscita.

Al contrario del chip 74154, non esistono ingressi G1 o G2 al chip 74L42 che possono essere usati per abilitare o disabilitare il chip stesso. Perciò, dovrete fornire $\overline{OUT}$ e $\overline{IN}$ lungo le uscite del canale decodificato se volete generare impulsi di selezione dispositivo.

Un solo chip 7402 vi permette di generare quattro impulsi di selezione dispositivo. Quattro di tali impulsi sono tutto ciò che vi occorre per la maggior parte degli esperimenti di questo Bugbook. Se volete, potete modificare i codici dei dispositivi di ingresso e uscita per farli corrispondere a quelli disponibili tamite l'uso di questo decodificatore. Un procedimento del genere può semplificare il lavoro di interfaccia per molti degli esperimento e dei programmi che si trovano in questo Bugbook.

**Programma**

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 074 | INR A | Incrementa il contenuto dell'accumulatore di 1 |
| 001 | 323 | OUT | Generazione di un impulso di selezione per il dispositivo di uscita 004 |
| 002 | 004 | 004 | Codice di selezione per il dispositivo 004 |
| 003 | 323 | OUT | Generazione dell'impulso di selezione dispositivo $\overline{\text{DS 005}}$ |
| 004 | 005 | 005 | Codice di dispositivo per $\overline{\text{DS 005}}$ |
| 005 | 323 | OUT | Generazione dell'impulso di selezione dispositivo $\overline{\text{DS 006}}$ |
| 006 | 006 | 006 | Codice di dispositivo per $\overline{\text{DS 006}}$ |
| 007 | 323 | OUT | Generazione dell'impulso di selezione dispositivo $\overline{\text{DS 007}}$ |
| 010 | 007 | 007 | Codice di dispositivo per $\overline{\text{DS 007}}$ |
| 011 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 012 | 000 | – | Byte d'indirizzo LO |
| 013 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Usando un solo chip 7402, montate il circuito mostrato nel grafico, nel quale, come si vede, vengono usati entrambi i segnali di controllo $\overline{\text{IN}}$ e $\overline{\text{OUT}}$. Caricate il programma nella memoria di lettura/scrittura.

**Passo 2**

Eseguite il programma a 750 kHz, poi operate avanzando di un passo.
Che cosa osservate sui quattro indicatori a LED?

Dovreste osservare che gli indicatori a LED associati con gli impulsi di selezione dispositivo OUT 006 e OUT 007 sono accesi, mentre gli altri due sono spenti 750 kHz. Questi ultimi due indicatori a LED sono accesi una volta ogni sedici cicli macchina, quando il programma viene fatto avanzare di un passo.

**Passo 3**

Perchè gli indicatori a LED per gli impulsi di selezione dispositivo IN 004 e IN 005 sono spenti?

Perchè non vi sono istruzioni IN nel programma!

**Passo 4**

Cambiate tutte le istruzioni OUT del programma in istruzioni IN. Usate il codice di istruzione 333, per IN. Eseguite il programma, ancora una volta a 750 kHz. Che cosa osservate?

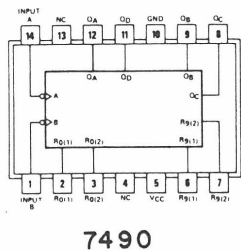
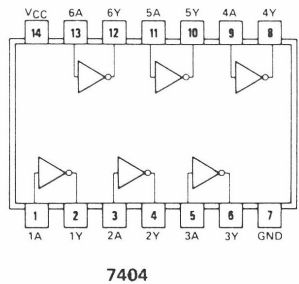
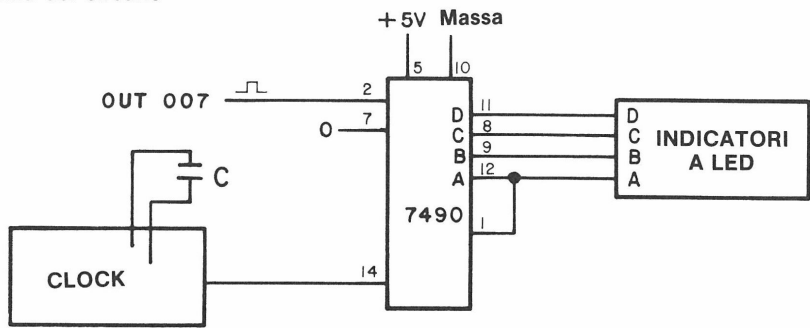
Ora gli indicatori a LED per gli impulsi IN 004 e IN 005 sono accesi, mentre quelli per OUT 006 e OUT 007 sono spenti. Le due istruzioni d'ingresso del programma generano due impulsi che sono rivelati dagli indicatori a LED. Gli impulsi di selezione dispositivo IN 006 e IN 007 non vengono decodificati dal circuito dato in questo esperimento.

*Lasciate questo esperimento come si trova e passate a quello successivo.*

## ESPERIMENTO N° 7

### Scopo

Questo esperimento dimostra l'uso di un impulso di selezione dispositivo per l'azzeramento di un contatore 7490.

### Configurazioni dei pin dei circuiti integrati

### Schema del circuito

### Programma

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 227 | SUB A | Azzera l'accumulatore |
| 001 | 323 | OUT | Generazione di un impulso di selezione per il dispositivo dato dal byte seguente |
| 002 | 007 | 007 | Codice dispositivo per l'impulso di selezione dispositivo DS 007. |

| | | | |
|---|---|---|---|
| 003 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 004 | 000 | – | Byte d'indirizzo LO |
| 005 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Per azzerare un contatore a decade 7490, avete bisogno di un impulso di selezione dispositivo positivo. Così, sarete in grado di usare l'impulso OUT 007 che avete prodotto nell'esperimento n° 6.

Montate il circuito mostrato nella figura. Caricate il programma suddetto nella memoria di lettura/scrittura. Assicuratevi che sia effettuato il collegamento OUT 007 fra l'uscita del gate 7402 e l'ingresso 7490 al pin 2.

**Passo 2**

Eseguite il programma a 750 kHz, poi operate avanzando di un passo. La frequenza di clock del contatore 7490 dovrebbe essere di circa 10 Hz. Abbiamo usato un capacità = di 0,05 μF per l'Outboard di clock.

**Passo 3**

Che cosa accade all'indicatore a LED?

Abbiamo osservato una cadenza di conteggio di 10 Hz sui LED del display fino al terzo ciclo macchina dell'istruzione OUT, durante il quale è stato generato un impulso di selezione dispositivo e il contatore è stato azzerato. Il conteggio è ripreso alla fine del terzo ciclo macchina.

**Passo 4**

L'istruzione 227, che azzera l'accumulatore, ha qualcosa a che fare con l'azzeramento del contatore 7490?

No! In questo programma, essa non ha alcuna influenza sul chip 7490, dal momento che non abbiamo effettuato nessun collegamento fra il bus dati bidirezionale, da D0 a D7, e il chip 7490. Di conseguenza, il 7490 non sa che l'accumulatore è stato azzerato.

*Potete togliere tutti i circuiti dalla piastra SK-10 eccetto il circuito monostabile e il circuito del contatore, che userete in un esperimento successivo.*

## ESPERIMENTO N. 8

### Scopo

Questo esperimento dimostra l'uso di due chip comparatori 7485 per decodificare in assoluto un indirizzo a 8 bit.

### Configurazioni dei pin dei circuiti integrati

7402

7485

### Schema del circuito

**Programma**

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 227 | SUB A | Azzera l'accumulatore |
| 001 | 323 | OUT | Generazione di un impulso di selezione dispositivo per il dispositivo dato dal byte seguente |
| 002 | 306 | 306 | Codice di selezione per il dispositivo di uscita 306 |
| 003 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 004 | 000 | – | Byte d'indirizzo LO |
| 005 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Userete il circuito mostrato nell'esperimento nº 4 di questo capitolo per contare gli impulsi di selezione dispositivo che sono prodotti dai due chip comparatori 7485. Che cambiamento dovete apportare al contatore 7490, conservato dall'esperimento precedente?

Ricollegare l'ingresso RESET del contatore 7490 (pin 2) al livello logico 0 (GND) e collegare l'ingresso CLOCK (pin 14) a OUT 306 sulla porta NOR a 2 ingressi 7402.

**Passo 2**

Montate il circuito mostrato nella figura e caricate il programma suddetto nella memoria di lettura/scrittura.

**Passo 3**

Eseguite il programma a 750 kHz, poi iniziate ad operare avanzando di un passo. Proseguite in questo modo l'esecuzione del programma, e spiegate nello spazio qui sotto che cosa osservate sul display a LED del contatore.

Abbiamo osservato un solo conteggio ogni volta che veniva eseguita l'istruzione OUT, o un conteggio ogni sette cicli macchina.

**Passo 4**

Ora cambiate il codice dispositivo da LO = 002 a LO = 305. Eseguite il programma a 750 kHz, poi operate ancora una volta avanzando di un passo.

Potete osservare il conteggio sui LED di uscita collegati al chip 7490?

No, perchè l'indirizzo generato dall'istruzione OUT non corrisponde più all'indirizzo predisposto al circuito comparatore.

**Passo 5**

L'indirizzo predefinito potrebbe essere cambiato per corrispondere ad un nuovo codice dispositivo software LO = 002?

Si.

Cambiate il byte d'indirizzo da LO = 002 a LO = 377. L'esecuzione del programma dà luogo a dei conteggi?

No.

Ora collegate di nuovo B0 fino a B8 sui chip comparatori in modo che siano tutti a livello logico 1. Questo fatto dà luogo a conteggi quando eseguite il programma? Perchè?

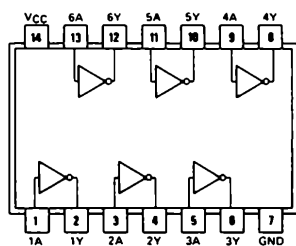
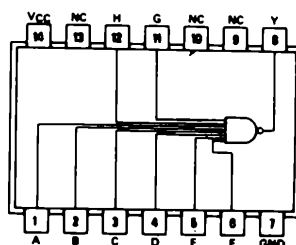
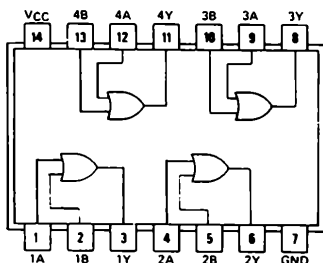
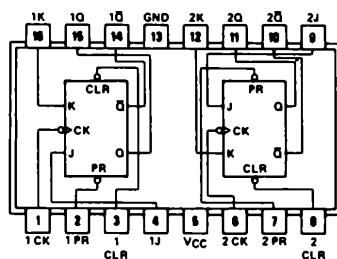
Sì. Ora l'indirizzo hardware predisposto e il codice dispositivo predisposto coincidono.

*Togliete il circuito decodificatore 7485 ma lasciate il contatore e i circuiti monostabili per l'esperimento successivo.*

## ESPERIMENTO N. 9

### Scopo

Questo esperimento dimostra l'uso di una porta NAND 7430 a 8 ingressi per decodificare in assoluto il byte del codice dispositivo di un bus d'indirizzo a 8 bit.

### Configurazioni dei pin dei circuiti integrati

7404 7430

7432 7476

### Programma

| Byte d'indirizzo LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 227 | SUB A | Azzera l'accumulatore |
| 001 | 323 | OUT | Generazione di un impulso di selezione dispositivo per predisporre il flip-flop 7476. |
| 002 | 306 | 306 | Codice dispositivo per l'ingresso predisposto del flip-flop 7476 |
| 003 | 323 | OUT | Generazione di un impulso di selezione dispositivo per azzerare il flip-flop 7476 |

**Schema del circuito**

| | | | |
|---|---|---|---|
| 004 | 317 | 317 | Codice dispositivo per l'ingresso di azzeramento del flip-flop 7476 |
| 005 | 303 | JMP | Salto incondizionato alla posizione di memoria data dai due byte d'indirizzo seguenti |
| 006 | 000 | – | Byte d'indirizzo LO |
| 007 | 003 | – | Byte d'indirizzo HI |

**Passo 1**

Se avete un relé a stato solido e una ventola o un altro dispositivo a corrente alternata appropriato, come una lampada, vorremmo incoraggiarvi a montare tutto il circuito mostrato precedentemente. Altrimenti, montate il circuito fino al buffer del relé a stato solido escluso. Usate un contatore 7490, come per l'esperimento n° 4, per contare gli impulsi in uscita dal flip-flop 7476.

**Passo 2**

Montate il circuito mostrato nella figura e caricate il programma suddetto in memoria.

**Passo 3**

Dopo aver iniziato l'esecuzione del programma a 750 kHz, proseguirete con il funzionamento passo a passo. Se state usando un relè a stato solido, potreste sospendere temporaneamente il collegamento fra il flip-flop 7476 e il buffer finché il programma avanza passo a passo. Perché?

La velocità più alta alla quale potete attivare e disattivare il relé è il doppio della frequenza di rete., o 120 volte al secondo. Se il microcomputer opera a 750 kHz con il programma suddetto, potrete attivare e disattivare il relé ad una velocità che è maggiore di 10.000 volte al secondo.

**Passo 4**

Spiegate che cosa osservate eseguendo il programma passo a passo.

Abbiamo osservato un conteggio singolo ogni volta che abbiamo eseguito un loop nel programma. Quando abbiamo cablato il circuito del relé a stato solido, abbiamo osservato che la ventola si accendeva quando eseguivamo l'istruzione che inizia con LO = 001. Quando eseguivamo l'istruzione OUT a LO = 003, la ventola si spegneva.

Abbiamo eseguito l'esperimento del relé a stato solido parecchie volte, ed ogni volta abbiamo osservato lo stesso risultato. Questo è un esperimento importante.

**Passo 5**

Che modifiche dovreste apportare al programma per eseguirlo a 750 kHz?

Avremo bisogno di almeno due loop di delay per dare al relé a stato solido il tempo sufficiente per accendersi e spegnersi. Un loop di delay verrebbe posizionato tra l'istruzione OUT 306 e l'istruzione OUT 317; l'altro verrebbe posizionato immediatamente dopo l'istruzione OUT 317, ma prima dell'istruzione OUT 306.

## DOMANDE RIEPILOGATIVE

Le seguenti domande vi aiuteranno a ripassare gli impulsi di selezione dispositivo.

1. Gli impulsi di selezione dispositivo possono essere usati per azzerare, fornire strobe, trigger, ecc. a circuiti integrati. Per i chip indicati di seguito, identificate l'esatto numero di pin al quale deve essere eseguita l'operazione indicata, e se occorre un impulso di selezione dispositivo positivo o negativo.

   a. Azzerare il primo flip-flop su di un chip 7474.

   b. Resettare i secondo flip-flop su di un chip 7474.

   c. Resettare a nove un contatore 7490.

   d. Azzerare un cantatore binario 7493.

   e. Abilitare un decodificatore 74154 da 4 a 16 linee

   f. Abilitare i primi due latch sul chip 7475.

   g. Fornire il clock al secondo flip-flop su di un chip 7474.

   h. Azzerare il primo monostabile su di un chip 74123.

   i. Fornire lo strobe al primo monostabile su di un chip 74123.

   j. Fornire un trigger al primo decodificatore di un chip 74155.

   k. Fornire un trigger al monostabile 74122.

2. Con il circuito di interfaccia mostrato nell'esperimento nº 4, è possibile generare sedici diversi impulsi di selezione dispositivo. Per quali codici dispositivo di uscita sarà generato un impulso di selezione dispositivo al canale 5 (pin 6) sul chip decodificatore 74154? State decodificando in assoluto il byte del codice dispositivo a 8 bit?

3. Quanti cicli macchina vi sono per le seguenti istruzioni 8080A?

   a. JNZ <B2> <B3>

   b. CALL <B2> <B3>

   c. OUT <B2>

   d. IN <B2>

   e. MOV C,M

   f. LXI H <B2> <B3>

   g. MVI B <B2>

   h. JMP <B2> <B3>

   i. INR A

   j. DCR B

**RISPOSTE**

1. a. Impulso di selezione dispositivo negativo al pin 1
   b. Impulso di selezione dispositivo negativo al pin 10
   c. Impulso di selezione dispositivo positivo al pin 7, con pin 6 a livello logico 0
   d. Impulso di selezione dispositivo positivo al pin 2
   e. Impulso di selezione dispositivo negativo al pin 18, col pin 19 a livello logico 0
   f. Impulso di selezione dispositivo positivo al pin 13
   g. Impulso di selezione dispositivo positivo al pin 11
   h. Impulso di selezione dispositivo negativo al pin 3
   i. Impulso di selezione dispositivo negativo al pin 1 con pin 2 a livello logico 1
   j. Impulso di selezione dispositivo negativo al pin 2
   k. Impulso di selezione dispositivo negativo al pin 1, coi pin 2,3 e 4 a livello logico 1

2. Un impulso di selezione dispositivo verrà prodotto per i seguenti byte di codice dispositivo di uscita (in codice ottale):

005
025.
045
065
105
125
145
165
205
225
245
265
305
325
345
365

Non state decodificando in assoluto il byte del codice dispositivo. Se lo faceste, l'unico codice dispositivo che fornirebbe un'uscita sul chip decodificatore 74154 sarebbe 005.

3. a. tre
   b. cinque
   c. tre
   d. tre
   e. due
   f. tre
   g. due
   h. tre
   i. uno
   j. uno

# CAPITOLO 18

# IL SET DI ISTRUZIONI 8080A

## INTRODUZIONE

Questo capitolo riassume tutte le caratteristiche importanti di ogni istruzione nel set di istruzioni dell'8080A: il numero dei cicli macchina, il numero di stati, il tipo di indirizzamento di memoria e i flag che vengono coinvolti dall'esecuzione dell'istruzione. Viene fornita una descrizione di ogni istruzione e, in alcuni casi, ne vengono dati esempi sull'uso. Alla fine del capitolo troverete inoltre parecchi esperimenti di programmazione.

## OBIETTIVI

Al termine di questo capitolo sarete in grado di:

- Indicare quali flag sono coinvolti nell'esecuzione di una data istruzione.
- Suddividere il set di istruzioni dell'8080A in cinque gruppi.
- Dare la definizione di contatore di programma, registro, accumulatore, registro universale, stack, stack pointer, registro istruzioni, codice istruzioni, coppia di registri, e "digit".
- Fare un elenco di molte sorgenti d'informazione per la programmazione dell'8080.
- Fare un elenco di diversi tipi di operazioni di trasferimento dati che avvengono all'interno di un microcomputer 8080A.
- Convertire un codice istruzioni a 8 bit in codice sia ottale che esadecimale con l'aiuto di una tabella fornita in questo capitolo.
- Distinguere fra condizionato e istruzione condizionata.
- Descrivere le caratteristiche dei cinque flag di condizione nel chip 8080A.
- Descrivere l'operazione dello stack, le istruzioni che ne influenzano il contenuto e la posizione dello stack stesso.
- Descrivere le quattro diverse istruzioni di rotazione dell'accumulatore
- Distinguere fra byte d'indirizzo LO e HI in istruzioni e programmi.

## LA PROGRAMMAZIONE CON I MICROCOMPUTER

A meno che non abbiate già una preparazione di base sui calcolatori o possediate un'abilità particolare nel programmare, probabilmente troverete che la programmazione a livello macchina ed a livello assembler sono qualcosa di noioso e inizialmente difficile.
Pare che non vi sia un modo più breve di imparare a programmare. Nel tempo dovuto, il vostro set di istruzioni e i trucchi della programmazione vi diventeranno abbastanza familiari da permettervi di scrivere modesti programmi con poco sforzo. Sarete in grado di applicare la vostra abilità, acquisita con un set di istruzioni, ad altri set di istruzioni, sia di microcomputer che di minicomputer e di grossi sistemi di elaborazione dati.

Coloro che sono interessati ai linguaggi ad alto livello, non avranno molto da aspettare. In aggiunta al package MIST BASIC, sono attesi un compilatore BASIC per l'8080 del Livermore Laboratory e un compilatore FORTRAN per l'8080 (Control Logic, Inc.).

Quello che vorremmo puntualizzare, comunque, è che senz'altro avrete bisogno di imparare a programmare in linguaggio assembler. I programmi semplici e le subroutines possono essere scritti facilmente e velocemente sia nel linguaggio assembler che in un liguaggio a più alto livello; tali programmi vengono eseguiti anche più velocemente, richiedono meno memoria e probabilmente sono più facili da capire. Dovrete imparare a programmare con il linguaggio assembler allo scopo di capire gli altri programmi assembler largamente usati, Infine, il saper programmare con il linguaggio assembler fornisce le basi per capire e confrontare altri set di istruzioni. Se qualcun altro programma per voi, vi costerà caro; se lo fate da soli sarà costoso ugualmente. Comunque, se potete adattare altri programmi alle vostre applicazioni, i vostri costi di programmazione saranno minori.

## FONTI D'INFORMAZIONE PER LA PROGRAMMAZIONE 8080

Vorremmo darvi un elenco di alcune fonti d'informazione riguardanti la programmazione 8080/8080A che consideriamo utili:

1. Intel Corporation, *Intel 8080 Microcomputer Systems User's Manual;* Intel Corporation, 3605 Bowers Avenue, Santa Clara, California 95051,.

   Il Capitolo 4 fornisce un sommario del set di istruzioni 8080/8080A. Per ogni tipo di istruzione, è elencato il numero di cicli macchina richiesto per eseguire l'istruzione. Se l'istruzione ha due possibili tempi di esecuzione, vengono elencati entrambi. Sono mostrati sia vari modi di indirizzare i dati significativi che i flag coinvolti nell'esecuzione dell'istruzione.

   In altri capitoli vengono discusse le funzioni di un computer, la CPU 8080, le tecniche di interfaccia dell'8080, e la famiglia 8080 dei componenti di supporto. Se lavorate seriamente con il microcomputer 8080, dovreste possedere questo manuale.

2. Intel Corporation, *Intel 8080 Assembler Language Programming Manual* 3605 Bowers Avenue, Santa Clara, California 95051.

   Un ottimo manuale che tratta argomenti quali il contatore di programma (program counter) lo stack pointer, la rappresentazione del programma in memoria, l'indirizzamento di memoria, i bit di condizione, il linguaggio assembler e tutto il set di istruzioni 8080.

Viene discusso anche l'uso delle istruzioni macro, che sono estremamente utili nella programmazione con il linguaggio assembler. Questo manuale è quello che vi occorrerà se programmate con il cross-assembler Intel 8080, o se leggete programmi che vengono assemblati usando il software Intel. Con l'iuto di questo manuale, si possono capire molti dei programmi della libreria Intel.

3. NEC Microcomputers, Inc., *The COM-8 Software Manual,* NEC Microcomputers, Inc., 5 Militia Drive, Lexington, Massachussets 02173.

Un eccellente manuale che presenta i seguenti problemi tipici della programmazione:

- Un semplice dispositivo di lettura
- Un contatore gated
- Un controllore programmato real time per un motore
- Una diramazione (branch) ad N vie nell'ambito di un programma
- Una subroutine d'interruzione
- Una subroutine di I/O per teletype
- Una subroutine di addizione o sottrazione BCD a 16 cifre
- Uno spostamento di dati nelle operazioni di memoria.
- Macroprogrammazione e assemblaggio condizionato.

Sono fornite ottime descrizioni sulle singole istruzioni 8080. Per ogni problema inerente la programmazione, vengono presentati diagrammi di flusso.

Questo manuale mostrerà alcune tecniche di programmazione molto utili a coloro che hanno già un'esperienza di programmazione con il linguaggio assembler 8080.

4. Intel Corporation, *Intel 8-bit User's Program Library,* Intel Corp. Unser's Library Microcomputer Systems, 3065 Bowers Avenue, Santa Clara, California 95051. È possibile diventare soci per un minimo di 12 mesi portando un contributo con un programma accettabile per la libreria, o pagando una quota associativa di $ 100.

I programmi sottoposti alla Libreria Utenti devono essere accompagnati dal *Microcomputer User's Library Submittal Form,* di cui viene fornita una copia alla fine di questo capitolo; le copie in grandezza naturale possono essere ordinate al Software Marketing Group alla Intel. Questo modulo viene usato dal direttore della libreria nel preparare il catalogo e gli aggiornamenti e la descrizione della "Funzione" è usata nella preparazione dell'indice del catalogo che viene mandato agli utenti futuri. Questo modulo è usato anche come prefisso di ogni programma contenuto nella libreria e dovrebbe perciò essere preparato attentamente. Sul retro del modulo vi sono istruzioni dettagliate a cui i programmi proposti dovrebbero strettamente attenersi. Questa documentazione standard viene mantenuta per garantire ad ogni membro interessato la possibilità di usare ogni programma della libreria.

Per sottoporre un programma alla Libreria, vi preghiamo di fare riferimento specialmente ai punti 1, 2, 3 e 4 delle istruzioni.
*Il programma non può essere un diplicato di un programma che esiste già nella Libreria.* Il programma dovrebbe essere senza errori e deve essere nel linguaggio Intel standard (4004, 4040, 8008, 8080, o PL/M). Accludete un elenco dattiloscritto delle fonti ed un nastro perforato.

Il package originale della Libreria Utenti venne aggiornato nel Dicembre 1975. Un secondo aggiornamento è stato effettuato il Settembre 1976 ed è stato poi aggiornato ogni due mesi. Fino al settembre 1976, vi erano 200 programmi nella libreria. È la più vasta libreria di programmi per qualunque microcomputer. A partire dal settembre 1977, pagando una piccola somma per l'uso, sono disponibili nastri sorgente.

Per informazioni sui programmi della User's Library, contattare direttamente la Intel Corporation.
Anche la National Semiconductor, seconda sorgente per il chip 8080, ha in sviluppo una User's Library.

5. Scelbi Computer Consulting Inc., 1322 Rear Boston Post Road, Milford, Connecticut 06460.

È disponibile il software seguente:

*La programmazione con il linguaggio macchina per l'8008 (e computer simili),*

*Un programma Assembler 8080*

*Un programma Editor 8080*

*Routines Monitor 8080*

*SCELBAL. Linguaggio base elementare scientifico per i sistemi 8008/8080.*

*SCELBI "8080" Software gowrmet guide cook book*

Nat Wadsworth scrive bene. Potete scegliere molte tecniche di programmazione con il microcomputer fra quelle proposte.

6. Zilog Coporation, *Z80-CPU Technical Manual,* Zilog, Inc., 170 State Street, Los Altos, California 94022.

Da questo manuale non ricaverete molti suggerimenti per la programmazione, ma è molto interessante confrontare il chip Z80 con l'8080A per quanto riguarda il set di istruzioni.

7. BYTE, Byte Publications, Inc., 70 Main Street, Peterborough, New Hampshire, 03458, $ 12 per due anni, $ 30 per tre anni.

La qualità dei singoli articoli varia, ma in questa rivista troverete programmi e tecniche di programmazione utili. La rivista è molto interessante e tratta il microcomputer come hobby.

8. National Semiconductor, *PACE Logic Designers Guide to Programmed Equivalents to TTL Functions,* National Semiconductor Corporation 2900 Semiconductor Drive, Santa Clara, California 95051.

Anche se serve per un microporocessore completamente diverso, il PACE a 16 bit, questo libro dà un'ottima dimostrazione di come si possa sostituire l'hardware con il software. Vengono forniti e descritti circuiti hardware. Vengono poi presentati programmi che duplicano le funzioni di base dell'hardware. Conoscendo un po' il set di istruzioni del PACE, dovreste essere in grado di convertire i programmi nel linguaggio Intel 8080. I vantaggi delle operazioni a 16 bit sono sicuramente evidenti.

9. 73 Magazine, Peterborough, New Hampshire 03458, $ 10 all'anno.

La rivista, che è diretta a radio-amatori, ha una parte di 40 pagine intitolate "I/O" che è dedicata all'uso pratico dei microcomputer. Dato che i dilettanti sono molto interessati alle comunicazioni, in questa rivista potreste trovare servizi sempre più numerosi sulle comunicazioni di dati digitali.

## SOMMARIO DEL SET DI ISTRUZIONI DELL'8080

I sommari del linguaggio assembler e del linguaggio macchina del set di istruzioni 8080 si possono ricavare da fonti diverse:

1. Intel Corporation, *Intel 8080 Assembly Language Reference Card,* Intel Corp., 3065 Bowers Avenue, Santa Clara, California 95051.

   Fornisce sia un elenco esadecimale del set di istruzioni 8080 che un elenco per funzione delle istruzioni. Vi è anche un elenco esadecimale ASCII.

2. Tychon, Inc., *8080 Octal Code Card,* Tychon, Inc., P.O. Box 242, Blacksburg, Virginia 24060.

   Un inserto scorrevole vi permette di trovare rapidamente il codice di istruzione ottale a 8 bit per un'istruzione in linguaggio assembler. È indicato anche lo stato del flag dopo un'istruzione.

3. Martin Research, *8080 Instruction Set,* 3336 Commercila Ave., Northbrook, Ill. 60062.

   Suddivide il set di istruzioni 8080 per funzioni.

4. R. Baker, *Byte,* 84 (Febbraio 1976).

   Elenco in codice ottale compatto del set di istruzioni dell'8080.

5. P.R. Rony, D.G. Larsen, e J.A. Titus, *Bugbook 8080*

   Il libro è edito in Italia da Jackson Editrice e offre una mole imponente di informazioni utili.

## I REGISTRI DEL MICROPROCESSORE 8080

Il termine *"registro"* può essere definito come segue:

| | |
|---|---|
| *Registro* | Un circuito di memoria elettronico digitale la cui capacità solitamente è di una parola.[15] |

I singoli registri del microprocessore 8080 memorizzano un solo byte, cioè otto bit contigui.

Nel chip 8080 vi sono due diversi tipi di registri: quelli che si possono e quelli che *non* si possono indirizzare da programma.
I primi sono mostrati nella figura della pagina seguente e comprendono:

- Sei registri universali a 8 bit indirizzati singolarmente o a coppia.

  *registro B*
  *registro C*
  *registro D*
  *registro E*
  *registro H*
  *registro L*

- *L'accumulatore* a 8 bit, noto anche come *registro A*
- *il registro stack pointer* a 16 bit
- il *registro contatore di programma* a 16 bit

Altri due registri sui quali, in casi speciali, potete avere dei controlli:

- *il registro istruzioni* a 8 bit
- *un registro flag* a 5 bit nell'unità aritmetico/logica (ALU).

I registri aggiuntivi che sono necessari per fare sì che il microprocessore 8080 esegua le sue operazioni interne sono: due registri temporanei a 8 bit usati singolarmente o due alla volta, il registro temporaneo *W e il registro temporaneo Z; un accumulatore temporaneo* a 8 bit nell'unità aritmetico/logica; ed un *registro temporaneo* a 8 bit nell'unità aritmetico/logica. Non potete indirizzare o controllare il contenuto di questi registri temporanei nel programma e non saprete quando vengono impiegati dall'8080.

Ecco alcune definizioni utili:

| | |
|---|---|
| *Contatore di programma (Program counter)* | Il registro a 16 bit del microprocessore 8080 che contiene l'indirizzo di memoria del byte di istruzioni successivo che dev'essere eseguito in un programma. |
| *Accumulatore* | Il registro e l'insieme dei circuiti elettronici digitali associati nell'unità aritmetico/logica (ALU) di un calcolatore nel quale vengono eseguite operazioni aritmetiche e logiche. |
| *Registri universali (General purpose)* | Nel microprocessore 8080A, i registri a 8 bit che possono collaborare alle operazioni aritmetiche e logiche con il contenuto dell'accumulatore. |

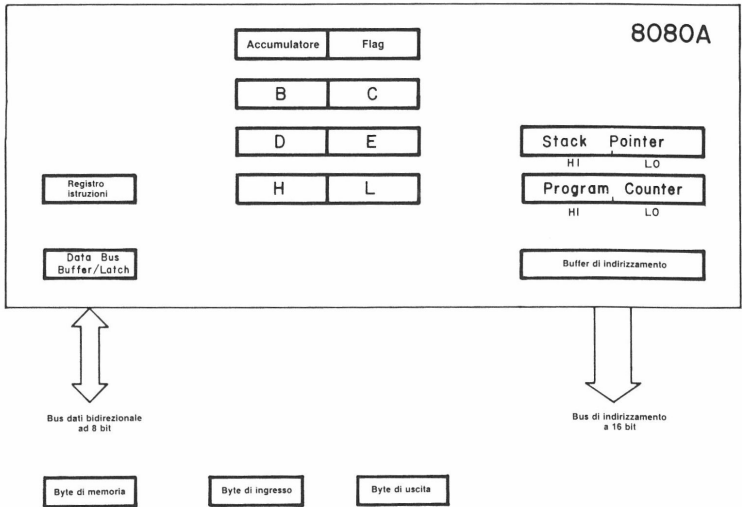



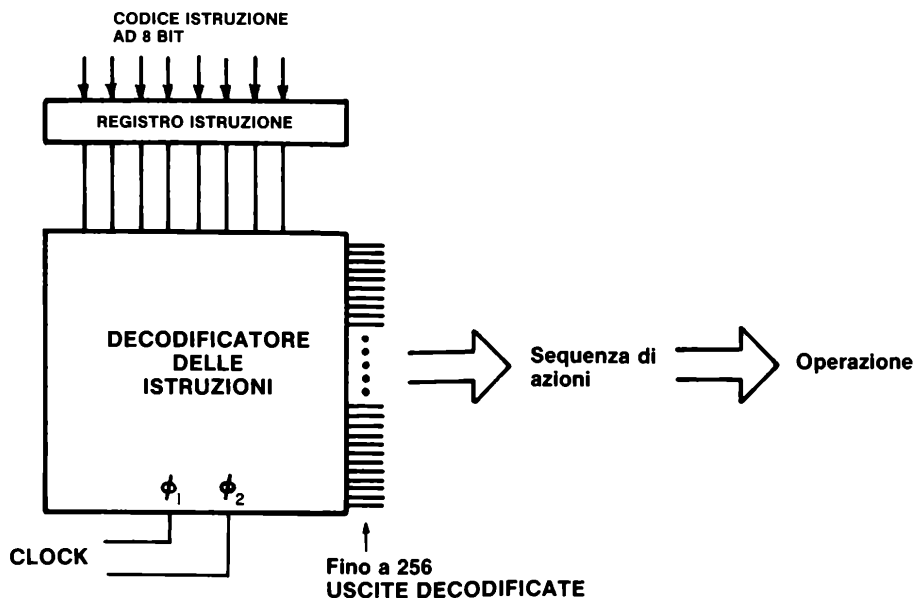
*Figura 18-1.* L'architettura dei registri nel microprocessore 8080. Sono stati omessi i registri temporanei sui quali non avete controllo diretto. Il buffer/latch del bus di dati e il buffer d'indirizzo fanno da interfaccia tra l'insieme dei circuiti interni al chip e i bus esterni.

**Schema a blocchi funzionale della CPU 8080A**

$D_7 \cdot D_0$ BI-DIRECTIONAL DATA BUS
DATA BUS BUFFER/LATCH
(8 BIT) INTERNAL DATA BUS
(8 BIT) INTERNAL DATA BUS
ACCUMULATOR (8)
TEMP REG (8)
FLAG (5) FLIP FLOPS
ACCUMULATOR LATCH (8)
ARITHMETIC LOGIC UNIT (ALU) (8)
DECIMAL ADJUST
INSTRUCTION REGISTER (8)
INSTRUCTION DECODER AND MACHINE CYCLE ENCODING
MULTIPLEXER
REGISTER SELECT
W (8) TEMP REG
Z (8) TEMP REG
B (8) REG
C (8) REG
D (8) REG
E (8) REG
H (8) REG
L (8) REG
STACK POINTER (16)
PROGRAM COUNTER (16)
INCREMENTER/DECREMENTER ADDRESS LATCH (16)
REGISTER ARRAY
TIMING AND CONTROL
POWER SUPPLIES +12V +5V -5V GND
WRITE
DATA BUS CONTROL
INTERRUPT CONTROL
HOLD CONTROL
WAIT CONTROL
SYNC
CLOCKS
$\overline{WR}$ DBIN INTE INT HOLD HOLD ACK WAIT READY SYNC $\phi 1$ $\phi 2$ RESET
ADDRESS BUFFER (16)
$A_{15} \cdot A_0$ ADDRESS BUS

*Figura 18-2.* Schema a blocchi funzionale dell'unità centrale dell'8080A (CPU). Notate il bus di dati interno che comunica con il bus di dati bidirezionale esterno per mezzo di un buffer/latch del bus di dati interno al chip 8080A.

| | |
|---|---|
| *Stack pointer* | Il registro a 16 bit del microprocessore 8080A che memorizza l'indirizzo di memoria della parte superiore dello stack che è una zona della memoria che memorizza le informazioni temporanee. |
| *Registro istruzione* | Il registro a 8 bit del microprocessore 8080A che memorizza il codice istruzioni dell'istruzione che è in esecuzione. |
| *Decodificatore delle istruzioni* | Un decodificatore all'interno del microprocessore 8080A che decodifica il codice istruzioni in una serie di azioni che vengono eseguite dal microprocessore. |
| *Codice istruzioni* | Un unico numero binario a 8 bit che codifica un'operazione che può essere eseguita dal microprocessore 8080A |

L'*Intellec 8/Mod 80 Microcomputer Development System Reference Manual* della Intel Corporation presenta parecchi paragrafi che riassumono i concetti di: codice istruzioni, registro dell'istruzione e docodificatore di istruzioni. Citiamo questi paragrafi più avanti. Anche la figura che segue dovrebbe esservi di aiuto.

"Ogni calcolatore ha una *lunghezza di parola* che è caratteristica di quella macchina. Nella maggior parte dei sistemi a otto bit, rende di più trattare campi binari a otto bit e la memoria associata ad un processore di questo tipo viene quindi organizzata per memorizzare otto bit in ogni posizione di memoria indirizzabile. I dati e le istruzioni vengono memorizzati come numeri binari di otto bit o come numeri che sono multipli interi di otto bit: 16 bit, 24 bit, e così via. Talvolta ci si riferisce a questo campo caratteristico a otto bit come ad un *"byte."*"

"Ogni operazione che il processore può eseguire viene identificata da un unico numero binario conosciuto come *"codice istruzioni"*. Una parola a otto bit usata come un codice istruzioni può distinguere fra 256 azioni alternative, più che adeguate per la maggior parte dei processori".

"Il processore *preleva dalla memoria (to fetch)* un'istruzione in due operazioni distinte. Nella prima, esso trasmette l'indirizzo alla memoria nel suo contatore di programma. Nella seconda, la memoria ridà al processore il byte indirizzato. La CPU memorizza questo byte di istruzioni in un registro noto come il *"registro istruzioni"*, e lo usa per dirigere le attività durante il ciclo di istruzioni restante."

"Il meccanismo attraverso il quale il processore traduce un codice istruzioni in specifici processi di elaborazione richiede un'attenzione maggiore di quella che possiamo prestare in questo contesto. Il concetto, comunque, sarà intuitivamente chiaro ad un progettista che abbia esperienza in fatto di logica. Gli otto bit memorizzati nel registro istruzioni possono essere decodificati ed usati per attivare in modo selettivo una delle linee di uscita, in questo caso 256. Ogni linea rappresenta un'insieme di attività associate all'esecuzione di un particolare codice istruzioni. La linea abilitata può essere combinata in coincidenza con gli impulsi di tempo selezionati per sviluppare segnali in sequenza elettrica che possono essere usati per dare il via ad azioni specifiche. Questa traduzione del codice in azione è eseguita dal *decodificatore di istruzioni* e dall'insieme dei circuiti di controllo associati."

Il punto importante da rilevare è che il codice istruzioni è tradotto in una sequenza di azioni specifiche. Il clock a due fasi è di importanza vitale per questo processo. Le azioni consistono nello spostare i dati dalla memoria nell'accumulatore, o nell'aggiungere il contenuto del registro B al registro A, o nel complementare l'accumulatore, o in una qualunque delle operazioni specifiche contenute nel set di istruzioni. Tuttavia, *ogni operazione specifica eseguita da un'istruzione 8080A è il risultato di una o più azioni specifiche causate dal decodificatore di istruzioni.*

## CHE TIPI DI OPERAZIONI ESEGUE IL MICROPROCESSORE 8080A?

Lo scopo di questo capitolo non è quello di suddividere il set di istruzioni 8080A in categorie, ma piuttosto di identificare i tipi fondamentali di operazioni che il chip attualmente esegue.

- SPOSTAMENTO DI UN BYTE DA UNA POSIZIONE AD UN'ALTRA

  Da un registro universale ad un altro
  Da un registro universale alla memoria, e *viceversa*
  Dall'accumulatore alla memoria, e *viceversa*

Dall'accumulatore a un registro universale, e *viceversa*
Dalla memoria al registro istruzioni
Dalla memoria al contatore di programma, e *viceversa*
Dalla memoria allo stack pointer
Dall'accumulatore ad un latch di uscita
Da un dispositivo d'ingresso all'accumulatore
Da un buffer esterno three-state al registro istruzioni
Dal flag register alla memoria, e *viceversa*
Da un registro universale allo stack pointer
Dal contatore di programma allo stack, e *viceversa*
Dai registri universali allo stack, e *viceversa*
Dall'accumulatore allo stack, e *viceversa*
Dal registro flag allo stack,e *viceversa*
Da un dispositivo d'ingresso al registro universale
Dal registro universale ad un dispositivo di uscita
Da un registro universale al contatore di programma

- OPERAZIONI ARITMETICHE E LOGICHE

AND dei contenuti del registro o della memoria con l'accumulatore
OR dei contenuti del registro o della memoria con l'accumulatore
OR Esclusivo dei contenuti del registro o della memoria con l'accumulatore
Confronto del contenuto del registro o della memoria con l'accumulatore
Somma del contenuto del registro o della memoria all'accumulatore (con o senza riporto)
Sottrazione del contenuto del registro o della memoria dall'accumulatore (con o senza riporto negativo)
Rotazione del contenuto dell'accumulatore
Incremento del contenuto del registro universale, della coppia di registri, dell'accumulatore, della memoria o dello stack pointer
Decremento del contenuto del registro universale, della coppia di registri, dell'accumulatore, della memoria o dello stack pointer
Somma del contenuto della coppia di registri al contenuto di un'altra coppia di registri o allo stack pointer
Aggiustamento decimale del contenuto dell'accumulatore

- OPERAZIONI MISTE

Nessuna operazione
Alt
Abilitazione del sistema di interruzione
Disabilitazione del sistema di interruzione
Complementazione dell'accumulatore
Set del flag di riporto
Complementazione del flag di riporto

Per la maggior parte del tempo, tutto quello che fa il microprocessore è spostare un byte da una posizione ad un altra e eseguire operazioni aritmetiche e logiche. Raramente, esegue una delle operazioni miste. In altre parole, il chip non calcola; sposta i byte.

## CODICE MNEMONICO DELLE ISTRUZIONI DELL'8080

Vi incoraggiamo ad imparare il più presto possibile i codici mnemonici, in modo che possiate programmare con il linguaggio assembler, leggere altri programmi in linguaggio assembler per l'8080 e migliorare la vostra capacità di capire i set di istruzioni per altri microprocessori. I codici mnemonici delle istruzioni dell'8080 sono elencati in gruppi nel *"Intel 8080 Microcomputer Systems User's Manual"*, che vi raccomandiamo di procurarvi. Qui farete prima un elenco dei codici mnemonici in ordine alfabetico e poi li descriverete dettagliatamente. Questo materiale proviene da due fonti d'informazione:

> *Intel 8080 Microcomputer Systems User's Manual*, Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051, 1975.
>
> *The µCOM-8 Software Manual*, NEC Microcomputers, Inc., Five Militia Drive, Lexington, Massachussets 02173, 1975.

Siamo grati del permesso accordatoci di usare le fonti suddette.

| | Codice istruzioni | | |
|---|---|---|---|
| Codice Mnemonico | Ottale | Esadecimale | Descrizione |
| ACI <B2> | 316 | CE | Somma del byte immediato all'accumulatore (con riporto) |
| ADC M | 216 | 8E | Somma del contenuto della memoria all'accumulatore (con riporto) |
| ADC r | 21*S* | † | Somma del contenuto del registro all'accumulatore (con riporto) |
| ADD M | 206 | 86 | Somma del contenuto della memoria all'accumulatore |
| ADD r | 20*S* | † | Somma del contenuto del registro all'accumulatore |
| ADI <B2> | 306 | C6 | Somma del byte immediato all'accumulatore |
| ANA M | 246 | A6 | AND del contenuto di memoria con l'accumulatore |
| ANA r | 24*S* | † | AND del contenuto del registro con l'accumulatore |
| ANI <B2> | 346 | E6 | AND del byte immediato con l'accumulatore |
| CALL <B2> <B3> | 315 | CD | Richiamo di subroutine in modo incondizionato |
| CC <B2> <B3> | 334 | DC | Richiamo di subroutine solo se il flag di riporto è settato |
| CM <B2> <B3> | 374 | FC | Richiamo di subroutine solo se il flag di segno è settato |

| | | | |
|---|---|---|---|
| CMA | 057 | 2F | Complemento del contenuto dell'accumulatore |
| CMC | 077 | 3F | Complemento del flag di carry |
| CMP M | 276 | BE | Confronto tra il contenuto della memoria e quello dell'accumulatore |
| CMP r | 27S | † | Confronto tra il contenuto del registro e quello dell'accumulatore |
| CNC <B2> <B3> | 324 | D4 | Richiamo di subroutine solo se il flag di carry è resettato |
| CNZ <B2> <B3> | 304 | C4 | Richiamo di subroutine solo se il flag di zero è resettato |
| CP <B2> <B3> | 364 | F4 | Richiamo di subroutine solo se il flag di segno è resettato |
| CPE <B2> <B3> | 354 | EC | Richiamo di subroutine solo se il flag di parità è settato |
| CPI <B2> | 376 | FE | Confronto in modo immediato tra il byte dato ed il contenuto dell'accumulatore |
| CPO <B2> <B3> | 344 | E4 | Richiamo di subroutine solo se il flag di parità è resettato |
| CZ <B2> <B3> | 314 | CC | Richiamo di subroutine solo se il flag di zero è settato |
| DAA | 047 | 27 | Aggiustamento decimale del contenuto dell'accumulatore |
| DAD B | 011 | 09 | Somma della coppia di registri B alla coppia di registri H |
| DAD D | 031 | 19 | Somma della coppia di registri D alla coppia di registri H |
| DAD H | 051 | 29 | Somma della coppia di registri H alla coppia di registri H |
| DAD SP | 071 | 39 | Somma della coppia di registri H allo stack pointer |
| DCR M | 065 | 35 | Decremento del contenuto di memoria |
| DCR r | 0D5 | † | Decremento del contenuto del registro |
| DCX B | 013 | 0B | Decremento del contenuto della coppia di registri B |
| DCX D | 033 | 1B | Decremento del contenuto della coppia di registri D |
| DCX H | 053 | 2B | Decremento del contenuto della coppia di registri H |
| DCX SP | 073 | 3B | Decremento dello stack pointer |
| DI | 363 | F3 | Disabilitazione del sistema d'interruzione |

| | | | |
|---|---|---|---|
| EI | 373 | FB | Abilitazione del sistema d'interruzione |
| HLT | 166 | 76 | Alt incondizionato |
| IN <B2> | 333 | DB | Ingresso dati nell'accumulatore |
| INR M | 064 | 34 | Incremento del contenuto di memoria |
| INR r | 0*D*4 | † | Incremento del contenuto del registro |
| INX B | 003 | 03 | Incremento dei contenuti della coppia di registri B |
| INX D | 023 | 13 | Incremento dei contenuti della coppia di registri D |
| INX H | 043 | 23 | Incremento dei contenuti della coppia di registri H |
| INX SP | 063 | 33 | Incremento dello stack pointer |
| JC <B2> <B3> | 332 | DA | Salto se il flag di carry è settato |
| JM <B2> <B3> | 372 | FA | Salto se il flag di segno è settato |
| JMP <B2> <B3> | 303 | C3 | Salto incondizionato |
| JNC <B2> <B3> | 322 | D2 | Salto se il flag di carry è resettato |
| JNZ <B2> <B3> | 302 | C2 | Salto se il flag di zero è resettato |
| JP <B2> <B3> | 362 | F2 | Salto se il flag di segno è resettato |
| JPE <B2> <B3> | 352 | EA | Salto se il flag di parità e settato |
| JPO <B2> <B3> | 342 | E2 | Salto se il flag di parità è resettato |
| JZ <B2> <B3> | 312 | CA | Salto se il flag di zero è settato |
| LDA <B2> <B3> | 072 | 3A | Caricamento diretto dell'accumulatore con il contenuto della locazione di memoria indirizzata da <B2> <B3> |
| LDAX B | 012 | 0A | Caricamento indiretto dell'accumulatore con il contenuto della locazione di memoria indirizzata dalla coppia di registri B |
| LDAX D | 032 | 1A | Caricamento indiretto dell'accumulatore con il contenuto della locazione di memoria indirizzata dalla coppia di registri D |
| LHLD <B2> <B3> | 052 | 2A | Caricamento di L ed H con i contenuti delle locazioni di memoria M ed M+1 |
| LXI B <B2> <B3> | 001 | 01 | Caricamento immediato della coppia di registri B |
| LXI D <B2> <B3> | 021 | 11 | Caricamento immediato della coppia di registri D |
| LXI H <B2> <B3> | 041 | 21 | Caricamento immediato della coppia di registri H |
| LXI SP <B2> <B3> | 061 | 31 | Caricamento immediato dello stack pointer |

| | | | |
|---|---|---|---|
| MVI M <B2> | 066 | 36 | Trasferimento immediato di un byte in una locazione di memoria |
| MVI r <B2> | 0*D*6 | † | Trasferimento immediato di un byte in un registro |
| MOV M,r | 16*S* | † | Trasferimento in una locazione di memoria del contenuto di un registro |
| MOV r,M | 1*D*6 | † | Trasferimento in un registro del contenuto di una locazione di memoria |
| MOV r1,r2 | 1*DS* | † | Trasferimento nel registro 1 del contenuto del registro 2 |
| NOP | 000 | 00 | Nessuna operazione |
| ORA M | 266 | B6 | OR del contenuto di una locazione di memoria con il contenuto dello accumulatore |
| ORA r | 26*S* | † | OR del contenuto di un registro con il contenuto dell'accumulatore |
| ORI <B2> | 366 | F6 | OR immediato di un byte con il contenuto dell'accumulatore |
| OUT <B2> | 323 | D3 | Uscita del contenuto dell'accumulatore |
| PCHL | 351 | E9 | Caricamento del Program Counter con il contenuto della coppia di registri H (salto indiretto) |
| POP B | 301 | C1 | Prelievo dallo Stack della coppia di registri B |
| POP D | 321 | D1 | Prelievo dello Stack della coppia di registri D |
| POP H | 341 | E1 | Prelievo dallo Stack della coppia di registri H |
| POP PSW | 361 | F1 | Prelievo dallo Stack del Program Status Word (flag ed accumulatore). |
| PUSH B | 305 | C5 | Inserimento nello stack della coppia di registri B |
| PUSH D | 325 | D5 | Inserimento nello stack della coppia di registri D |
| PUSH H | 345 | E5 | Inserimento nello stack della coppia di registri H |
| PUSH PSW | 365 | F5 | Inserimento nello stack della Program Status Word (flag ed accumulatore) |
| RAL | 027 | 17 | Rotazione a sinistra del contenuto dell'accumulatore attraverso il Carry |
| RAR | 037 | 1F | Rotazione a destra del contenuto dell'accumulatore attraverso il carry |
| RC | 330 | D8 | Ritorno se il flag di carry è settato |
| RET | 311 | C9 | Ritorno incondizionato |
| RLC | 007 | 07 | Rotazione a sinistra del contenuto dell'accumulatore |
| RM | 370 | F8 | Ritorno se il flag di segno è settato |
| RNC | 320 | D0 | Ritorno se il flag di carry è resettato |
| RNZ | 300 | C0 | Ritorno se il flag di zero è resettato |
| RP | 360 | F0 | Ritorno se il flag di segno è resettato |
| RPE | 350 | E8 | Ritorno se il flag di parità è settato |
| RPO | 340 | E0 | Ritorno se il flag di parità è resettato |

| | | | |
|---|---|---|---|
| RRC | 017 | 0F | Rotazione a destra del contenuto dell'accumulatore |
| RST n | 3*N*7 | † | Richiamo della subroutine alle locazioni HI=000, LO=On |
| RZ | 310 | C8 | Ritorno se il flag di zero è settato |
| SBB M | 236 | 9E | Sottrazione dall'accumulatore del contenuto della locazione di memoria indicata (con riporto negativo) |
| SBB r | 23*S* | † | Sottrazione dall'accumulatore del contenuto del registro indicato (con riporto negativo) |
| SBI <B2> | 336 | DE | Sottrazione immediata di un byte dall'accumulatore (con riporto negativo) |
| SHLD <B2> <B3> | 042 | 22 | Memorizzazione in M, M+1 del contenuto della coppia di registri H (M=< B2>, <B3>) |
| SPHL | 371 | F9 | Trasferimento nello stack pointer del contenuto della coppia di registri H |
| STA <B2> <B3> | 062 | 32 | Memorizzazione del contenuto dell'accumulatore direttamente nella locazione di memoria data da <B2> , <B3> |
| STAX B | 002 | 02 | Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri B |
| STAX D | 022 | 12 | Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D |
| STC | 067 | 37 | Set del flag di carry |
| SUB M | 226 | 96 | Sottrazione del contenuto di memoria dall'accumulatore |
| SUB r | 22*S* | † | Sottrazione del contenuto del registro dall'accumulatore |
| SUI <B2> | 326 | D6 | Sottrazione immediata di un byte dall'accumulatore |
| XCHG | 353 | EB | Scambio dei contenuti della coppia di registri D e H |
| XRA M | 256 | AE | OR-esclusivo memoria-accumulatore |
| XRA r | 25*S* | † | OR- esclusivo registro-accumulatore |
| XRI <B2> | 356 | EE | OR-esclusivo immediato di un byte con l'accumulatore |
| XTHL | 343 | E3 | Scambio del top dello stack con il contenuto della coppia di registri H |

Non tutti i 256 possibili codici istruzione ($2^8=256$) sono utilizzati dall'8080. Alcuni dei codici non usati sono i seguenti:

| | |
|---|---|
| 010 | 08 |
| 020 | 10 |

| | |
|---|---|
| 030 | 18 |
| 040 | 20 |
| 050 | 28 |
| 060 | 30 |
| 070 | 38 |
| 313 | CB |
| 331 | D9 |
| 335 | DD |
| 355 | ED |
| 375 | FD |

Nota: le istruzioni aventi la notazione "+", non sono facilmente traducibili in codice esadecimale senza informazioni su registri usati, od altro. Questa è una delle ragioni per cui abbiamo scelto i numeri ottali. Procederemo nella descrizione del set di istruzioni dell'8080 in dettaglio.

Useremo materiale ricavato sia dall'*"Intel 8080 Microcomputer System User's Manual"* che da *"The µCOM-8 Software Manual"*, per concessione della Intel Corporation e della NEC Microcomputers Inc.
Le istruzioni sono raggruppate come segue:

- GRUPPO TRASFERIMENTO DATI: spcstamento di dati tra i registri o tra la memoria ed i registri.
- GRUPPO ARITMETICO: somma, sottrazione, incremento o decremento nei contenuti dei registri o nella memoria.
- GRUPPO LOGICO: AND, OR, OR-ESCLUSIVO, confronto, rotazione o complementazione dei dati dei registri nella memoria.
- GRUPPO DI SALTO: istruzioni di salto condizionato ed incondizionato, richiamo di subroutine e ritorno di istruzioni.
- GRUPPO STACK, I/O, CONTROLLO MACCHINA: istruzioni di I/O, gestione dello stack e controllo interno dei flag.

## SET DI ISTRUZIONI

Un computer, per quanto "raffinato", può fare solo quello che gli viene "detto" di fare. Si dice al computer cosa fare attraverso una serie di istruzioni codificate a cui ci si riferisce come ad un **Programma.** Il regno del programmatore è costituito dal **Software,** al contrario dell'**Hardware** che è compreso negli attuali computer. Il software di un computer fa riferimento a tutti i programmi che sono stati scritti per quel computer.

Nella progettazione di un computer, il tecnico fornisce alla CPU la capacità di eseguire un particolare set di operazioni. La CPU è realizzata in modo tale che viene eseguita un'operazione specifica quando la logica di controllo della CPU stessa decodifica una particolare istruzione. Di conseguenza, le operazioni che possono essere eseguite dalla CPU definiscono il **Set di Istruzioni** del computer.

Ogni istruzione permette al programmatore di iniziare l'esecuzione di una specifica operazione. Tutti i computer implementano determinate operazioni aritmetiche nel loro set di istruzioni, come un'istruzione di somma del contenuto di due registri. Spesso sono comprese nel set di istruzioni operazione logiche (ad esempio OR del contenuto di due registri) e istruzioni operative su un registro (ad esempio incremento di un registro). Il set di istruzioni di un computer avrà anche istruzioni che spostano i dati fra i registri, fra un registro e la memoria e fra un registro e un dispositivo di I/O. La maggior parte dei set di istruzioni fornisce anche le **Istruzioni Condizionali.** Un'istruzione condizionale specifica un'operazione che va eseguita solo se si sono verificate determinate condizioni; per esempio, saltare ad una particolare istruzione se il risultato dell'ultima operazione era zero. Le istruzioni condizionali forniscono un programma in grado di "prendere decisioni".

Organizzando in modo logico una sequenza di istruzioni all'interno di un programma coerente, il programmatore può "dire" al computer di eseguire una funzione specifica e molto utile.

Il computer, comunque, può eseguire solo programmi le cui istruzioni sono in forma di codice binario (ad es. una serie di 1 e 0), che viene chiamato **Codice Macchina.** Dato che sarebbe estremamente scomodo programmare in codice macchina, sono stati sviluppati i linguaggi di programmazione. Sono disponibili programmi che convertono le istruzioni scritte in linguaggio di programmazione, in codice macchina, che può venire interpretato dal processore.

Un tipo di linguaggio per programmare è il **Linguaggio Assembler.** Ad ognuna delle istruzioni del computer viene assegnato un solo codice mnemonico del linguaggio assembler. Il programmatore può scrivere un programma (chiamato **Programma Sorgente)** usando questi codici mnemonici e determinati operandi; il programma sorgente viene poi convertito in istruzioni macchina (chiamate il **Codice Oggetto).** Ogni istruzione del linguaggio assembler viene poi convertita in un'istruzione di codice macchina (1 o più byte) da un programma **Assembler.** I linguaggi assembler dipendono solitamente dalla macchina (ed essi di solito sono in grado di funzionare solo su di un tipo di computer).

**IL SET DI ISTRUZIONI DEL MICROPROCESSORE 8080**

Il set di istruzioni dell'8080 comprende cinque diversi tipi di istruzioni:

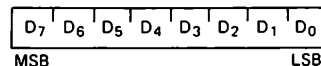
- **Gruppo Trasferimento Dati** - sposta i dati fra un registro e l'altro o fra la memoria e i registri
- **Gruppo Aritmetico** - addiziona, sottrae, incrementa o decrementa i dati nei registri o nella memoria
- **Gruppo Logico** - AND, OR, OR ESCLUSIVO, confronta, fa ruotare o complementa i dati nei registri o nella memoria
- **Gruppo Branch** - istruzioni di salto condizionato e incondizionato, istruzioni di chiamata della subroutine e istruzioni di rientro
- **Gruppo Stack, I/O e Controllo Macchina** - comprende le istruzioni I/O, nonchè le istruzioni per mantenere lo stack e i flag di controllo interni.

### Formato delle Istruzioni e dei Dati:

La memoria per l'8080 è organizzata in quantità di 8 bit, chiamati byte. Ogni byte ha un solo indirizzo binario a 16 bit che corrisponde alla sua posizione sequenziale in memoria.

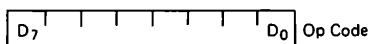
L'8080 può indirizzare direttamente fino a 65.536 byte di memoria, che possono essere costituiti sia da elementi di memoria di sola lettura (ROM) che da elementi di memoria ad accesso casuale (RAM) (memoria di lettura/scrittura).

I dati nell'8080 vengono memorizzati sotto forma di numeri interni binari a 8 bit:

PAROLA DATI

Quando un registro o una parola dati contiene un numero binario, è necessario stabilire l'ordine nel quale sono scritti i bit del numero. Nell'8080, ci si riferisce al BIT 0 come al **bit meno significativo (LSB)** e al BIT 7 (di un numero di 8 bit) come al **bit più significativo (MSB).**

Le istruzioni di programma dell'8080 possono essere una, due o tre byte in lunghezza. Le istruzioni a più byte devono essere memorizzate in successive locazioni di memoria; l'indirizzo del primo byte viene sempre usato come indirizzo delle istruzioni. L'esatto formato dell'istruzione dipenderà dalla particolare operazione che deve essere eseguita.

Istruzioni ad un byte

$D_7$ ... $D_0$ Op Code

Istruzioni a due byte

Istruzioni a tre byte

## Modi di Indirizzamento

Spesso i dati sui quali bisogna operare sono immagazzinati in memoria. Quando si usano dati numerici a più byte, i dati, come le istruzioni, vengono memorizzati in posizioni di memoria successive, con il byte meno significativo al primo posto, seguito via via dai byte più significativi. L'8080 ha quattro modi diversi di indirizzare i dati caricati in memoria o nei registri:

- Diretto - I byte 2 e 3 dell'istruzione contengono l'esatto indirizzo di memoria dei dati in questione (i bit d'indirizzo di ordine inferiore sono nel byte 2, i bit di ordine più elevato sono nel byte 3).
- Registro - L'istruzione specifica il registro o la coppia di registri in cui i dati sono posizionati.
- Registro Indiretto - L'istruzione specifica una coppia di registri che contiene l'indirizzo di memoria in cui i dati sono posizionati (i bit di ordine più elevato sono nel primo dei due indirizzi, i bit di ordine inferiore nel secondo).
- Immediato- L'istruzione contiene i dati stessi. Questi sono in quantità o di 8 o di 16 bit (il byte meno significativo per primo, il byte più significativo per secondo).

A meno che non venga diretta da un'istruzione d'interruzione o di salto, l'esecuzione delle istruzioni procede attraverso posizioni di memoria che aumentano sequezialmente. Un'istruzione di salto può specificare l'indirizzo dell'istruzione successiva che deve essere eseguita in uno dei due modi seguenti:

- Diretto - L'istruzione di salto contiene l'indirizzo dell'istruzione seguente che va eseguita (Eccetto che per l'istruzione 'RST' il byte 2 contiene l'indirizzo di ordine inferiore e il byte 3 l'indirizzo di ordine più elevato).
- Registro Indiretto - L'istruzione di salto indica una coppia di registri che contiene l'indirizzo dell'istruzione che va eseguita successivamente. (I bit d'indirizzo di ordine più elevato sono nel primo dei due registri, i bit di ordine inferiore nel secondo).

L'istruzione RST è una speciale istruzione di richiamo ad un byte (usata di solito durante le sequenze d'interruzione). RST comprende un campo a 3 bit; il controllo del programma viene trasferito all'istruzione il cui indirizzo è otto volte il contenuto di questo campo a tre bit.

## Flag di Condizione (Condition Flags):

Ci sono cinque flag di condizione collegati con l'esecuzione delle istruzioni sull'8080. Essi sono Zero, Sign, Parity, Carry, e Auxiliary Carry, ed ognuno di loro è rappresentato da un registro ad 1 bit nella CPU. Un flag viene "settato" forzando il bit verso 1, "resettato" forzando il bit verso 0.

Salvo diversa indicazione, quando un'istruzione coivolge un flag lo fa nel modo seguente:

Zero: Se il risultato di un'istruzione ha il valore 0, questo flag è settato; altrimenti viene resettato.

Sign: Se il bit più significativo del risultato dell'operazione ha il valore 1, questo flag è settato; altrimenti viene resettato.

Parity: Se la somma modulo 2 dei bit del risultato dell'operazione è 0, (ad es. se il risultato ha parità even), questo flag viene settato; altrimenti viene resettato (ad es. se il risultato ha parità odd).

Carry: Se l'istruzione determina un riporto (da addizione), o un riporto negativo (da una sottrazione o da una comparazione) fuori dal bit più significativo, questo flag viene settato; altrimenti viene resettato.

| | |
|---|---|
| Auxiliary Carry: | Se l'istruzione ha dato luogo ad un riporto dal bit 3 al bit 4 del valore risultante, il riporto ausiliario viene settato; altrimenti, è resettato. Questo flag riguarda le addizioni, le sottrazioni, gli incrementi, i decrementi, le comparazioni e le operazioni logiche, ma viene usato principalmente con le addizioni e gli incrementi che precedono un'istruzione DAA (Decimal Adjust Accumulator) |

## Simboli ed Abbreviazioni

I simboli e le abbreviazioni che seguono vengono usati nella descrizione successiva delle istruzioni dell'8080:

| SIMBOLI | SIGNIFICATO |
|---|---|
| accumulatore | registro A |
| addr | quantità d'indirizzo a 16 bit |
| data | quantità di dati a 8 bit |
| data 16 | quantità di dati a 16 bit |
| byte 2 | il secondo byte dell'istruzione |
| byte 3 | il terzo byte dell'istruzione |
| port | indirizzo a 8 bit di un dispositivo I/O |
| r, r1, r2 | uno dei registri A,B,C,D,E,H,L |
| DDD,SSS | la configurazione di bit che designa uno dei registri A,B,C,D,E,H,L (DDD = destinazione, SSS = sorgente): |

| DDD o SSS | NOME DEL REGISTRO |
|---|---|
| 111 | A |
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |

| | |
|---|---|
| rp | una delle coppie di registri: B rappresenta la coppia B,C dove B è il registro di ordine più elevato e C il registro di ordine inferiore; D rappresenta la coppia D,E dove D è il registro di ordine più elevato ed E il registro di ordine inferiore; H rappresenta la coppia H,L dove H è il registro di ordine più elevato e ed L il registro di ordine inferiore SP rappresenta il registro stack pointer a 16 bit |
| RP | La configurazione di bit che designa una delle coppie di registri B,D,H,SP: |

| RP | COPPIA DI REGISTRI |
|---|---|
| 00 | B-C |
| 01 | D-E |
| 10 | H-L |
| 11 | SP |

| | |
|---|---|
| rh | Il primo registro (ordine più elevato) di una coppia di registri designata. |
| rl | Il secondo registro (ordine inferiore) di una coppia di registri designata. |
| PC | Registro contatore di programma a 16 bit (PCH e PCL vengono usati per riferirsi rispettivamente agli 8 bit di ordine più elevato e a quelli di ordine inferiore. |
| SP | Registro stack pointer a 16 bit (SPH e SPL vengono usati per riferirsi rispettivamente agli 8 bit di ordine più elevato e a quelli di ordine inferiore). |
| rm | Bit m del registro r (i bit vanno dal numero 7 allo 0 da sinistra verso destra). |
| Z, S, P, CY, AC | I flag di condizione: Zero, Sign, Parity, Carry, e Auxiliary Carry, rispettivamente. |
| ( ) | Il contenuto della locazione di memoria o dei registri chiuso in parentesi |
| ← | "Viene trasferito a" |
| ∧ | AND logico |
| ⊻ | OR esclusivo |
| V | OR inclusivo |
| + | Addizione |
| − | Sottrazione di complemento di due |
| • | Moltiplicazione |
| ↔ | "Viene scambiato con'" |
| ‾‾‾ | Il complemento ad uno (es. (A)) |
| n | Numeri da 0 a 7 |
| NNN | La rappresentazione binaria da 000 a 111 |

**Descrizione del Format:**

Le pagine seguenti presentano una descrizione dettagliata del set di istruzioni dell'8080. Ogni istruzione è descritta nel modo seguente:

1. Il formato assembler MAC 80, che consiste del codice mnemonico di istruzione e dei campi operandi, è stampato in **NERETTO** sul lato sinistro della prima riga.
2. Il nome dell'istruzione è chiuso fra parentesi sul lato destro della prima riga.
3. La riga o le righe successive contengono una descrizione simbolica dell'operazione dell'istruzione.
4. Segue quindi una descrizione dell'operazione dell'istruzione.
5. La riga o le righe seguenti contengono le configurazioni e i campi binari che comprendono l'istruzione macchina.

6. Le ultime quattro righe contengono informazioni incidentali circa l'esecuzione dell'istruzione. Per primo è elencato il numero di stati e di cicli macchina richiesto per eseguire l'istruzione. Se l'istruzione ha due tempi di esecuzione possibili, come nel Salto Condizionato, saranno indicati tutti e due i tempi, separati da un trattino. Quindi, sono mostrati tutti i modi significativi di indirizzamento dei dati . L'ultima riga indica alcuni dei cinque flag che sono coinvolti nell'esecuzione dell'istruzione.

## GRUPPO TRASFERIMENTO DATI

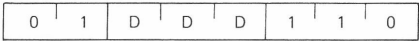
Questo gruppo di istruzioni trasferisce i dati da e verso i registri e la memoria. In questo gruppo i flag di condizione non vengono coinvolti da nessuna istruzione.

### MoV r1, r2

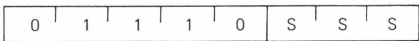
L'istruzione MOV r1, r2 trasferisce i dati dal registro sorgente S specifico ( r2) al registro destinazione D specifico ( r1). La sorgente o la destinazione possono essere uno qualunque dei singoli registri B, C, D, H, o L, l'accumulatore A, e M (il contenuto dell'indirizzo di memoria specificato dalla coppia di registri H, L). Nel byte a tre cifre ottali, la prima cifra è sempre un 1. La seconda e la terza cifra ottale variano a seconda della sorgente e della destinazione.
L'istruzione ottale, 166, è un'istruzione di arresto piuttosto che un'istruzione MOV. Il contenuto del registro sorgente non cambia durante un'istruzione MOV; esso viene duplicato.

### MOV r,M

L'istruzione MOV r,M trasferisce i dati da M (il contenuto dell'indirizzo di memoria specificato dalla coppia di registri H,L) al registro destinazione D specifico, che può essere uno qualunque dei singoli registri B, C, D, H, o L o l'accumulatore A. Il contenuto dell'indirzzo di memoria viene duplicato nel registro; il contenuto della memoria resta invariato.

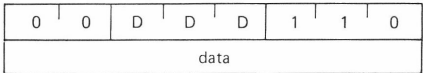
**MOV r,M** (Spostare dalla memoria)
(r) ← (H) (L)
Il contenuto della locazione di memoria, il cui indirizzo è nei registri H ed L, viene spostato nel registro r.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

## MOV M,r

**MOV M,r** (Spostare verso la memoria)
((H) (L)) ← (r)
Il contenuto del registro r viene spostato verso la locazione di memoria il cui indirizzo è nei registri H e L.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione MOV M,r trasferisce i dati dal registro sorgente S specificato, ad M (l'indirizzo di memoria specificato dalla coppia di registri H,L). Il registro sorgente può essere uno qualunque dei singoli registri B, C, D, E, H, o L o l'accumulatore A. I contenuti del registro vengono duplicati in memoria; i contenuti del registro restano invariati.

## MVI r, data

MVI r, data (Spostare in modo immediato)
(r) ← (byte 2)
Il contenuto del byte 2 dell'istruzione viene spostato verso il registro r.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: nessuno

L'istruzione MVI r, data trasferisce i dati dal secondo byte dell'istruzione a due byte al registro destinazione D specificato (r). Il termine *"immediato"* si riferisce al fatto che il byte di dati è contenuto all'interno dell'istruzione a più byte. Il registro destinazione specificato può essere uno dei singoli registri B, C, D, E, H o L, o l'accumulatore A, e M (i contenuti dell'indirizzo di memoria specificato dalla coppia di registri H,L). Quando la destinazione è M, avete l'istruzione MVI M, data, trattata qui sotto. I dati possono essere costituiti da qualunque numero binario a 8 bit fra 00000000 e 11111111.

## MVI M, data

**MVI M, data** (Spostare in modo immediato verso la memoria)
(H) (L) ← (byte 2)
Il contenuto del byte 2 dell'istruzione viene spostato nella locazione di memoria il cui indirizzo è nei registri H ed L.

Cicli: 3
Stati: 10
Indirizzamento: immediato/reg. indiretto
Flag: nessuno

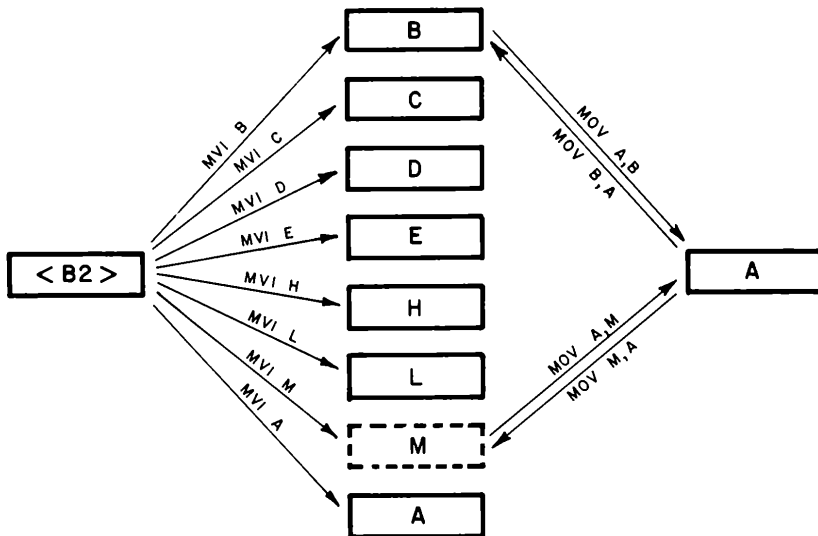
L'istruzione MVI M, data trasferisce i dati dal secondo byte dell'istruzione ad M (l'indirizzo di memoria specificato dalla coppia di registri H,L). I dati possono essere costituiti da qualunque numero binario a 8 bit fra 00000000 e 11111111.

## LXI rp,data 16

**LXI rp,data 16** (Caricare in modo immediato la coppia di registri)
(rh) ← (byte 3),
(rl) ← (byte 2)
Il byte 3 dell'istruzione viene spostato nel registro di ordine più elevato (rh) della coppia di registri rp. Il byte 2 dell'istruzione viene spostato nel registro di ordine inferiore (rl) della coppia di registri rp.

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

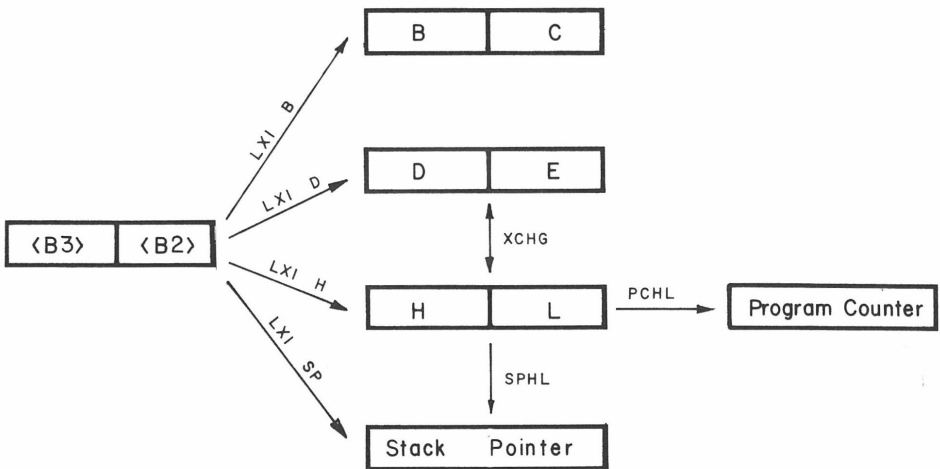
L'istruzione LXI rp, data fa sì che una quantità di dati a 16 bit contenuta nel secondo e nel terzo byte dell'istruzione venga caricata nella coppia di registri specificata da RP. RP può essere una qualunque delle coppie di registri HL, DE o BC o lo stack pointer, che sono rappresentati dai codici mnemonici H, D, B e SP, rispettivamente.
Il secondo byte dell'istruzione viene caricato nei registri LO - L, E, C, o negli 8 bit LO dello stack pointer; il terzo byte dell'istruzione è caricato nei registri HI - H, D, B o negli otto bit HI dello stack pointer. La parola dati a 16 bit può variare da 0000000000000000 a 1111111111111111, in notazione binaria.

Gli schemi seguenti illustrano alcune delle caratteristiche delle istruzioni MOV, MVI e LXI. Vengono mostrati solo due set delle istruzioni MOV r1, r2.

Da notare che LXI rp, data è equivalente a due istruzioni MVI r,data. Così,

LXI B
<B2>
<B3>

è equivalente a

MVI B
<B2> *(corrisponde a B3 nell'istruzione LXI B)*
MVI C
<B2> *(corrisponde a B2 nell'istruzione LXI B)*

In un'istruzione a due byte, il secondo byte è indicato sempre come <B2>. Una sola istruzione LXI rp, data richiede 10 stati per essere eseguita, mentre due istruzioni MVI r,data richiedono in tutto un tempo di esecuzione di 14 stati. Perciò, usando l'istruzione LXI rp,data, risparmiate 4 stati del tempo di esecuzione. In molti casi, un tale risparmio non ha importanza.

## STA addr

**STA addr** (Memorizzare direttamente l'accumulatore)
( (byte 3) (byte 2) ) ← (A)
Il contenuto dell'accumulatore viene spostato nella locazione di memoria il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione.

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

L'istruzione STA addr vi permette di memorizzare i contenuti dell'accumulatore direttamente nella locazione di memoria senza usare la coppia di registri H,L. L'indirizzo della locazione di memoria è specificato nel secondo e nel terzo byte dell'istruzione. Il byte d'indirizzo LO è il byte 2 e il byte d'indirizzo HI è il byte 3.

## LDA addr

**LDA addr** (Caricare direttamente l'accumulatore)
(A) ← ( (byte 3) (byte 2) )
Il contenuto della locazione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato al registro A.

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

L'istruzione LDA addr vi permette di caricare l'accumulatore con i contenuti della locazione di memoria specificata dai byte <B2> e <B3> nell'istruzione. Non avete bisogno di usare la coppia di registri H,L. Il byte d'indirizzo LO è <B2> e il byte d'indirizzo HI è <B3>.

## LHLD addr

**LHLD addr** (Caricare direttamente H ed L)
(L) ← ( (byte 3) (byte 2) )
(H) ← ( (byte 3) (byte 2) + 1)
Il contenuto della locazione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato al registro L. Il contenuto della locazione di memoria all'indirizzo successivo viene spostato al registro H.

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

Questa istruzione è utile quando le locazioni di memoria contengono l'informazione d'indirizzo.

In questo caso LHLD addr fa sì che il registro H venga caricato con il byte di memoria indirizzato dai byte <B2> e <B3> nell'istruzione, es. addr. Il registro H viene caricato con il byte di memoria posizionato a addr + 1. Si esegue così un trasferimento a 16 bit di un indirizzo di memoria alla coppia di registri H,L. Una volta studiato XCHG, osserverete che la sezione di programma.

```
LHLD
<B2>
<B3>
XCHG
```

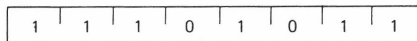
è funzionalmente equivalente a

```
LXI H
<B2>
<B3>
MOV E,M
INX H
MOV D,M
```

La prima sezione richiede 20 stati per l'esecuzione; la seconda sezione richiede 29 stati.

**XCHG**

**XCHG** (Scambiare H ed L con D e E)
(H) ←→ (D)
(L) ←→ (E)
I contenuti dei registri H ed L vengono scambiati con i contenuti dei registri D ed E.

| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: nessuno

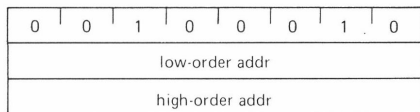
L'istruzione XCHG fa sì che i contenuti delle coppie di registri D,E e H,L vengano scambiati fra loro. Per essere precisi, si scambiano i contenuti dei registri D e H e dei registri E ed L. Questa istruzione vi permette di usare la coppia di registri H,L come un indirizzo di memoria, mentre un altro indirizzo di memoria viene conservato nella coppia di registri D,E. Potete modificare i contenuti della coppia di registri D,E senza cambiare la coppia di registri H,L. Per esempio, la coppia di registri H,L può specificare una posizione di memoria che usate per modificare la coppia di registri D,E. Due istruzioni XCHG in sequenza,

```
XCHG
XCHG
```

equivalgono ad una "no operation."

## SHLD addr

**SHLD addr** (Memorizzare direttamente H ed L)
( (byte 3) (byte 2) ) ← (L)
( (byte 3) (byte 2) + 1) ← (H)
Il contenuto del registro L viene spostato nella locazione di memoria il cui indirizzo è specificato nei byte 2 e 3. Il contenuto del registro H viene spostato nella posizione di memoria successiva.

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

L'istruzione SHLD addr fa sì che i contenuti del registro L vengano memorizzati nella locazione di memoria data dai byte <B2> e <B3> nell'istruzione, es. addr. I contenuti del registro H vengono memorizzati nella locazione di memoria, addr + 1. In altre parole, si esegue un trasferimento a 16 bit di 2 byte d'indirizzo della coppia di registri H,L verso le due locazioni di memoria successive, addr + 1. Questa istruzione è utile nel creare un gruppo di locazioni di memoria che contengono informazioni sull'indirizzo piuttosto che dati. Come con la maggior parte delle istruzioni dell'8080A, il byte <B2> è il byte d'indirizzo LO e il byte <B3> è il byte d'indirizzo HI addr.

La sezione di programma

```
XCHG
SHLD
<B2>
<B3>
```

è equivalente alla sezione di programma,

```
LXI H
<B2>
<B3>
MOV M,E
INX H
MOV M,D
```

## LDAX rp

L'istruzione LDAX rp vi permette di caricare l'accumulatore con i contenuti della locazione di memoria indirizzata da una coppia di registri piuttosto che dalla coppia di registri H,L. Così, con LDAX B, usate la coppia di registri B,C per fornire l'indirizzo di memoria a 16 bit; con LDAX D, usate la coppia di registri D,E per fornire l'indirizzo. La sezione di programma

LXI D
<B2>
<B3>
LDAX D

carica l'accumulatore come

LXI H
<B2>
<B3>
MOV A,M

**LDAX rp** (Caricare indirettamente l'accumulatore)
(A) ← ((rp))
Il contenuto della locazione di memoria, il cui indirizzo è nella coppia di registri rp, viene spostato nel registro A. Nota: solo le coppie di registri rp=B (registri B e C) o rp = D (registri D ed E) possono essere specificate.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
flag: nessuno

## STAX rp

**STAX rp** (Memorizzare indirettamente l'accumulatore)
( (rp)) ← (A)
Il contenuto del registro A viene spostato nella locazione di memoria il cui indirizzo è nella coppia di registri rp. Nota: solo le coppie di registri rp=B (registri B e C) o rp=D (registri D ed E) possono essere specificate.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione STAX rp vi permette di memorizzare i contenuti dell'accumulatore nella locazione di memoria indirizzata sia dalla coppia di registri B,D che dalla coppia di registri D,E. La sezione di programma

LXI B
<B2>
<B3>
STAX B

attua lo store dell'accumulatore come

```
LXI H
<B2>
<B3>
MOV M,A
```

Il significato delle istruzioni STAX rp e LDAX rp è che avete tre indirizzi di memoria a 16 bit indipendenti memorizzati nei registri universali all'interno del microprocessore 8080A. Sono disponibili sufficienti istruzioni da permettervi di usare tutti e tre gli indirizzi.

I flag di condizione non sono coinvolti da nessuna delle istruzioni della lista seguente:

```
MOV r1,r2
MOV r,M
MOV M,r
MVI r,data
MVI M,data
LXI rp, data 16
STA addr
LDA addr
XCHG
LHLD addr
SHLD addr
LDAX rp
STAX rp
```

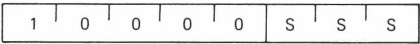
Queste istruzioni costituiscono il gruppo trasferimento dati nel microprocessore 8080A.

## GRUPPO ARITMETICO

Questo gruppo di istruzioni esegue operazioni aritmetiche su dati nei registri e nella memoria. *Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Sign, Parity, Carry e Auxiliary Carry secondo le regole standard.* Tutte le operazioni di sottrazione sono eseguite per mezzo dell'aritmetica complemento a due e posizionano il riporto ad uno per indicare un riporto negativo, e lo azzerano se non vi è nessun riporto negativo.

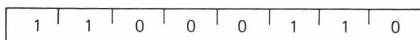
### ADD r

**ADD r** (Addizionare il registro)
(A) ← (A) + (r)
Il contenuto del registro r è addizionato al contenuto dell'accumulatore. Il risultato è posto nell'accumulatore.

| 1 | 0 | 0 | 0 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento: registro
flag: Z,S,P,CY,AC

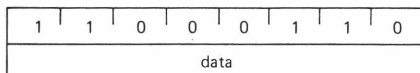
L'istruzione ADD r fa sì che i contenuti del registro sorgente S vengano addizionati ai contenuti dell'accumulatore. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H,L, l'accumulatore A, o M (i contenuti della memoria indirizzati dalla coppia di registri H,L). L'istruzione ADD M è descritta qui sotto. L'istruzione coinvolge tutti e quattro i flag testabili: Carry, Parity, Zero, e Sign. È coinvolto anche l'Auxiliary Carry.

## ADD M

**ADD M** (Addizionare la memoria)
(A) ← (A) + ((H) (L))
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L è addizionato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

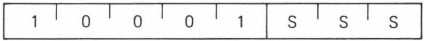
L'istruzione ADD M fa sì che i contenuti della locazione di memoria M, che è indirizzata dalla coppia di registri H,L, vengano addizionati ai contenuti dell'accumulatore. I contenuti della memoria restano invariati dopo l'addizione. L'istruzione coinvolge tutti e cinque i flag ed ha due cicli macchina.

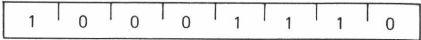
## ADI data

**ADI data** (Addizionare immediatamente)
(A) ← (A) + (byte 2)
Il contenuto del secondo byte dell'istruzione è addizionato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ADI data fa sì che i dati presenti nel secondo byte dell'istruzione vengano addizionati ai contenuti dell'accumulatore. L'istruzione coinvolge tutti e cinque i flag.

## ADC r e ADC M

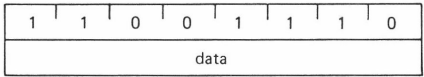
**ADC r** (Addizionare il registro con riporto)
(A) ← (A) + (r) + (CY)
Il contenuto del registro r e il contenuto del bit di riporto sono addizionati al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

**ADC M** (Addizionare la memoria con riporto)
(A) ← (A) + ((H) (L)) + (CY)
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L e il contenuto del flag CY sono addizionati all'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

Citiamo il *"μCOM-8 Software Manual"*: "Allo scopo di eseguire operazioni di addizione e sottrazione, occorrono delle speciali istruzioni aritmetiche. L'aritmetica a più cifre richiede che due elementi vengano visualizzati e conservati da qualche parte. Questi due elementi sono la somma delle cifre come sono state addizionate e la presenza o l'assenza di un bit di riporto. Quando viene prodotto un bit di riporto, esso deve essere addizionato alla somma delle cifre seguenti. In modo analogo, con le operazioni di sottrazione, l'esistenza di un riporto negativo dev'essere individuata, in modo che esso possa venire trattenuto dalla differenza delle cifre seguenti. Le istruzioni Addizione con Riporto e Sottrazione con Riporto Negativo fanno in modo che vengano visualizzati e conservati i bit di riporto, facendo addizioni e sottrazioni a più cifre in modo molto chiaro. ADC r, ADC M e ACI data sono le istruzioni "Addizione con Riporto". ADC r fa sì che i contenuti della sorgente S vengano addizionati alla somma dei contenuti dell'accumulatore e al bit di riporto."

Le istruzioni ADC r e ADC M sono simili alle istruzioni ADD r e ADD M. L'unica differenza è che il bit di riporto è addizionato al bit meno significativo nel byte dell'accumulatore. Tutti i flag sono coinvolti da queste istruzioni. La locazione di memoria M è indirizzata dai contenuti della coppia di registri H,L.

## ACI data

**ACI data** (Addizionare immediatamente con riporto)
(A) ← (A) + (byte 2) + (CY)
Il contenuto del secondo byte dell'istruzione e il contenuto del flag CY sono addizionati ai contenuti dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ACI data fa sì che la quantità di dati a 8 bit presente nel secondo byte dell'istruzione venga addizionata alla somma dei contenuti dell'accumulatore e al bit di riporto. L'istruzione coinvolge tutti e cinque i flag.

## SUB r e SUB M

**SUB r** (Sottrarre il registro)
(A) ← (A) − (r)
Il contenuto del registro r è sottratto dal contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

| 1 | 0 | 0 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

**SUB M** (Sottrarre la memoria)

(A) ← (A) − ((H) (L))
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L è sottratto dal contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione SUB r fa sì che i contenuti del registro sorgente S vengano sottratti dall'accumulatore. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H e L, l'accumulatore A, o M (i contenuti della memoria come vengono indirizzati dalla coppia di registri H,L). Tutti e cinque i flag sono coinvolti dall'esecuzione di questa istruzione. Se volete azzerare l'accumulatore, lo farà l'istruzione.

SUB A

che ha un codice istruzione di 227.

## SUI data

**SUI data** (Sottrarre in modo immediato)
(A) ← (A) − (byte 2)
Il contenuto del secondo byte dell'istruzione è sottratto dal contenuto dell'accummulatore. Il risultato si trova nell'accumulatore.

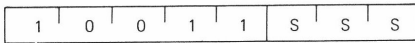
Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione SUI data fa sì che la quantità di dati a 8 bit specificata nel secondo byte d'istruzioni venga sottratta dall'accumulatore.

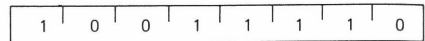
Tutti e cinque i flag sono coinvolti.

### SBB r e SBB M

**SBB r** (Sottrare il registro con riporto negativo)
(A) ← (A) − (r) − (CY)
Il contenuto del registro r e il contenuto del flag CY vengono sottratti entrambi dall'accumulatore.
Il risultato si trova nell'accumulatore.

| 1 | 0 | 0 | 1 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

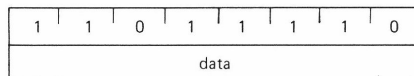
**SBB M** (Sottrarre la memoria con riporto negativo)
(A) ← (A) − ((H) (L)) − (CY)
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L e il contenuto del flag CY sono entrambi sottratti dall'accumulatore.
Il risultato si trova nell'accumulatore.

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione SBB r fa sì che i contenuti della sorgente S vengano sottratti dalla differenza dei contenuti dell'accumulatore e del bit di riporto negativo. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H e L; l'accumulatore A, o M, i contenuti di memoria indirizzati dalla coppia di registri H,L. Tutti e cinque i flag sono coinvolti dalle istruzioni SBB r e SBB/M.

### SBI data

**SBI data** (Sottrarre immediatamente con riporto negativo)
(A) ← (A) − (byte 2) − (CY)
I contenuti del secondo byte dell'istruzione e i contenuti del flag CY vengono entrambi sottratti dall'accumulatore. Il risultato si trova nell'accumulatore.

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione SBI data fa sì che la quantità di dati a 8 bit specificata nel secondo byte d'istruzioni venga sottratta dalla differenza dei contenuti dell'accumulatore e del bit di riporto negativo. Sono coinvolti tutti e cinque i flag.

È il caso di mostrare alcuni esempi delle varie operazioni di addizione e sottrazione. Considerate il programma seguente:

```
ADD B
ADD C
```

Se i contenuti iniziali del registro sono A = 00111110, B = 11100000 e C = 00101111, e se il bit di riporto fosse inizialmente zero, allora la sezione di codice suddetta produrrebbe il seguente risultato nell'accumulatore:

```
Bit di riporto

     0             00111110   Contenuti dell'accumulatore
                  +11100000   Contenuti del registro B
                   --------
     1             00011110   Somma memorizzata nell'accumulatore
                  +00101111   Contenuti del registro C
                   --------
     0             01001101   Somma memorizzata nell'accumulatore
```

Notate attentamente il comportamento del bit di riporto in questa situazione. Se non deriva nessun riporto dal bit più significativo (MSB) nell'accumulatore, il bit di riporto viene azzerato; se dal bit più significativo deriva un riporto nell'accumulatore durante l'addizione, il bit di riporto viene settato. Quando avete addizionato B all'accumulatore, avete avuto un riporto. Quando avete addizionato i contenuti di C alla somma, non c'è stato riporto. Il riporto di operazioni precedenti non viene utilizzato o "portato avanti". Adesso confrontiamo i risultati suddetti con il comportamento della seguente sezione di codice:

```
ADC B
ADC C
```

Supponiamo di avere gli stessi valori iniziali per i registri A, B, C e il bit di riporto. Otterreste i seguenti risultati:

```
Bit di riporto

     0             00111110   Contenuti dell'accumulatore
                  +11100000   Contenuti del registro B
                   --------
     1             00011110   Somma memorizzata nell'accumulatore
```

Fin qui, non c'è differenza. Comunque, quando addizioniamo i contenuti del registro C alla somma suddetta, osserviamo una differenza:

```
                   00011110   Somma memorizzata nell'accumulatore
                  +       1   Bit di riporto
                  +00101111   Contenuti del registro C
                   --------
     0             01001110   Somma memorizzata nell'accumulatore
```

Ora consideriamo la seguente sezione di programma

```
SUB B
SUB C
```

per gli stessi valori iniziali dei registri A, B, C, e il bit di riporto. Notate che se ricavate un riporto negativo dall'MSB dell'accumulatore, il bit di riporto viene settato; se non vi è nessun riporto negativo, il bit di riporto viene azzerato. Dovreste perciò osservare quanto segue:

| Bit di riporto | | |
|---|---|---|
| 0 | 00111110 | Contenuti dell'accumulatore |
| | − 11100000 | Contenuti del registro B |
| 1 | 01011110 | Differenza memorizzata nell'accum. |
| | − 00101111 | Contenuti del registro C |
| 0 | 00101111 | Differenza memorizzata nell'accum. |

Ora eseguiamo operazioni di sottrazione usando le istruzioni SSB r,

SBB B
SBB C

Abbiamo i seguenti risultati:

| Bit di riporto | | |
|---|---|---|
| 0 | 00111110 | Contenuti dell'accumulatore |
| | − 11100000 | Contenuti del registro B |
| 1 | 01011110 | Differenza memorizzata nell'accum. |

Quando eseguiamo l'operazione SBB C sottraiamo i contenuti del registro C dalla differenza fra il bit di riporto negativo e i contenuti dell'accumulatore:

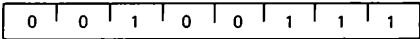
| | | |
|---|---|---|
| | 01011110 | Differenza memorizzata nell'accum. |
| | − 1 | |
| | − 00101111 | Contenuti del registro C |
| 0 | 00101110 | Differenza memorizzata nell'accum. |

Le istruzioni ADC r e SBB r vengono usate quando si eseguono operazioni aritmetiche in precisione doppie o triple. Un'operazione aritmetica in *doppia precisione* è un'operazione eseguita su due quantità di 16 bit per ottenere un risultato a 16 bit. Un'operazione in *tripla precisione* è quella eseguita su due quantità a 24 bit per ottenere un risultato a 24 bit. Gli esempi suddetti di operazioni di addizione e sottrazione sono stati gentilmente forniti dalla NEC Microcomputers, Inc. e sono tratti dal loro *"μCOM-8 Software Manual"*.

**DAA**

**DAA** (Aggiustamento decimale dell'accumulatore)
Il numero a otto bit nell'accumulatore viene adattato per formare cifre decimali in codice binario a quattro bit per mezzo del seguente procediemnto:

1. Se il valore dei 4 bit meno significativi dell'accumulatore è maggiore di 9 o se il flag AC è settato, si addiziona 6 all'accumulatore.
2. Se il valore dei 4 bit più significativi dell'accumulatore è ora maggiore di 9, o se il flag CY è settato, si addizione 6 ai 4 bit più significativi dell'accumulatore.

NOTA: Tutti i flag sono coinvolti.

Cicli: 1

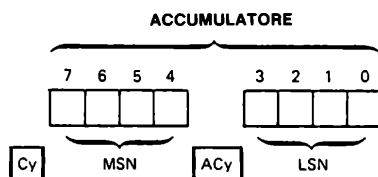
Stati: 4

Flag: Z,S,P,CY,AC

Citiamo il *"μCOM-8 Software Manual":* Allo scopo di eseguire operazioni decimali in codice binario (BCD), occorre un'istruzione speciale. Quando la CPU dell'8080A esegue un'operazione aritmetica, produce il risultato in codice binario. Lavorando in BCD, questo non produce il risultato corretto. Per rimediarvi, si usa un'istruzione DAA. DAA sta per Aggiustamento Decimale, che è esattamente ciò che fa DAA. L'istruzione DAA tratta l'accumulatore a 8 bit come due accumulatori a 4 bit. Tramite l'uso di un flag non testabile conosciuto come l'Auxiliary Carry (Riporto Ausiliario), l'operazione DAA adatta il risultato di un'operazione di addizione binaria al BCD impaccato .

"Per l'addizione, l'istruzione DAA dà luogo all'operazione seguente. Se il Carry Ausiliario è posizionato su uno o se il digit meno significativo (LSD) è maggiore di 9, si addiziona sei al digit meno significativo. Poi, se il flag di riporto è posizionato su uno o il digit più significativo è maggiore di 9, si addiziona sei al digit più significativo (MSD). Il termine "digit" viene cosí definito:

*Digit* — Un gruppo di quattro bit contigui che solitamente rappresentano una cifra decimale in codice binario (BCD).

Il digit meno significativo (LSD), il digit più significativo (MSD), l'accumulatore, il flag di riporto ausiliario (ACy)n e il flag di riporto (Cy) possono essere rappresentati nel modo seguente:

Supponiamo, come mostra un esempio del *"μCOM-8 Software Manual"*, che l'accumulatore contenga la rappresentazione BCD di 75 (MSD = 0111 e LSD = 0101) e che il registro B contenga la rappresentazione BCD di 38 (MSD = 0011 e LSD = 1000) e il flag di riporto sia a livello logico 0. L'istruzione, ABC B, produce il seguente risultato nell'accumulatore:

| Bit di riporto | Bit di riporto ausiliario | | |
|---|---|---|---|
| 0 | – | 0 1 1 1 0 1 0 1 | Contenuti dell'accumulatore |
| | | + 0 0 1 1 1 0 0 0 | Contenuti del registro B |
| 0 | 0 | 1 0 1 0 1 1 0 1 | Somma memorizzata nell'accumulatore |

Con il Riporto Ausiliario, se l'istruzione produce un riporto dal bit 3 nel bit 4 del valore risultante, il flag di Riporto Ausiliario è settato; altrimenti viene resettato. Nell'esempio precedente, non c'è riporto dal bit 3 e nel bit 4, e il bit di Riporto Ausiliario è zero dopo l'operazione.

Il comando DAA trova ACy resettato a 0 e LSDN = 1101. Dato che l'LSD è maggiore di 9, vi si addiziona sei e il risultato è 0011. Dato che l'MSD è maggiore di nove, vi si addiziona sei e il risultato è 0001. Il risultato finale dopo l'operazione DAA è

| | | | |
|---|---|---|---|
| 1 | 1 | 0 0 0 0 0 0 1 1 | Somma aggiustata decimale |
| 1 | – | 1 3 | Numero decimale |

che è equivalente al numero decimale. 103. L'operazione DAA può essere scritta nel modo seguente:

| Bit di riporto | Bit di riporto ausiliario | | | |
|---|---|---|---|---|
| 0 | 0 | 1 0 1 0 | 1 1 0 1 | Somma |
| | | + 0 1 1 0 | + 0 1 1 0 | Operazione DAA |
| 1 | 1 | 0 0 0 0 | 0 0 1 1 | Risultato dell'operazione DAA |
| 1 | 1 | 0 0 0 1 | 0 0 1 1 | BCD |
| 1 | – | 0 | 3 | Numero decimale |

Così, 75 + 38 = 103.

"Nell'operazione attuale, l'adattamento DAA è fatto in parallelo, piuttosto che in serie, come illustrato. Comunque, questa spiegazione seriale (ricavata dal *"µCOM-8 Software Manual"* della NEC Microcomputers, Inc.) è più semplice da capire ed illustra meglio l'adattamento. *L'istruzione DAA dovrebbe seguire immediatamente un'operazione di addizione o di incremento, dato che alcune istruzioni dell'8080A alterano lo stato del flag di riporto ausiliario. Un'alterazione di questo tipo potrebbe rivelarsi in risultati non corretti.".*

C'è una differenza importante fra il chip Intel 8080A e il chip equivalente, il chip µCOM-8 della NEC Microcomputers, Inc. Il chip µCOM-8 ha un flag non provabile in più, chiamato Subtract. Citiamo dal Manuale NEC: "Per l'addizione, il flag Sub viene posizionato su zero . . . . Per la sottrazione, Sub è posizionato su uno, dando luogo all'operazione DAA successiva. Se ACy è posizionato su uno (vi è stato un riporto negativo) si sottrae sei da LSD. Poi, se Cy è posizionato su uno (vi è stato un riporto negativo) si sottrae sei da MSD. L'uso di un'istruzione DAA immediatamente dopo un'operazione su due byte in formato BCD impaccato aggiusta il risultato a due cifre BCD e ad un riporto (sia negativo che positivo) in formato BCD
Notate che le operazioni DAA vengono eseguite direttamente dopo la sottrazione, *eliminando la necessità dell'aritmetica di complemento per 100 per la sottrazione".*

Se state eseguendo un notevole numero di manipolazioni BCD, dovrebbe interessarvi il chip COM-8 piuttosto che il chip 8080A. Comunque, ciò sarebbe valido nel caso aveste bisogno di far lavorare il microcomputer alla massima velocità. Con istruzioni aggiuntive, l'8080A può facilmente assolvere lo stesso compito di produrre un formato BCD dopo una sottrazione.

## INR r e INR M

L'istruzione INR r fa sì che un uno venga addizionato al registro di destinazione D. Il registro di destinazione può essere uno qualunque dei registri universali B, C, D, E, H, e L; l'accumulatore A; o M, i contenuti di memoria così come vengono indirizzati dalla coppia di registri H,L. Tutti i flag sono coinvolti, ad eccetto di quello di riporto (CY).

## DCR r e DCR M

**DCR r** (Decrementare il registro)
(r) ← (r) − 1
Il contenuto del registro r è decrementato di uno.
Nota: Tutti i flag, **eccetto CY,** sono coinvolti.

| 0 | 0 | D | D | D | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3
Stati: 5
Indirizzamento: reg. indiretto
Flag: Z,S,P,AC

**DCR M** (Decrementare la memoria)
( (H) (L) ) ← ( (H) (L) ) − 1
Il contenuto della locazione di memoria il cui indirizzo è contenuto nella coppia H ed L è decrementato di uno.
Nota: Tutti i flag, **eccetto CY,** sono coinvolti.

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: Z,S,P,AC

L'istruzione DCR r fa sì che venga sottratto un uno dal registro di destinazione D. Il registro di destinazione può essere uno qualunque dei registri universali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti di memoria così come vengono indirizzati dalla coppia di registri H, L. Solo quattro dei cinque flag sono coinvolti; il flag di riporto resta invariato.

## INX rp e DCX rp

**INX rp** (Incrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) + 1
Il contenuto della coppia di registri rp è incrementato di uno.
Nota: **Nessun flag è coinvolto.**

| 0 | 0 | R | P | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

**DCX rp** (Decrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) − 1
Il contenuto della coppia di registri rp è decrementato di uno.
Nota: **Nessun flag è coinvolto.**

| 0 | 0 | R | P | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

INX rp fa sì che la coppia di registri specificata da RP venga incrementata di uno; DCX rp fa sì che la coppia di registri specificata da RP venga decrementata di uno. RP può essere la coppia di registri specificata da B, D, o H (che corrispondono a BC, DE, o HL) o lo stack pointer a 16 bit specificato da SP. INX e DCX non coinvolgono nessun bit di flag; essi di solito non vengono usati nelle operazioni aritmetiche, dato che il loro uso principale sta nell'incrementare e decrementare gli indirizzi di memoria a 16 bit.

## DAD rp

**DAD rp** (Addizionare la coppia di registri ad H ed L)
(H) (L) ← (H) (L) + (rh) (rl)
Il contenuto della coppia di registri rp è addizionato al contenuto della coppia di registri H ed L.
Il risultato si trova nella coppia di registri H ed L.
Nota: **solo il flag CY è coinvolto.** Esso è settato se vi è un riporto dall'addizione in doppia precisione; altrimenti è resettato.

| 0 | 0 | R | P | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 6
Stati: 10
Indirizzamento: registro
Flag: CY

Secondo il Manuale NEC: "Mentre le istruzioni INX e DCR permettono l'incremento e il decremento della coppia di registri, l'istruzione DAD, doppio ADD, permette di addizionare insieme le coppie di registri. DAD rp fa sì che la coppia di registri specificata da RP venga addizionata ai contenuti della coppia di registri HL, e il risultato rimane nella coppia HL. Il flag di riporto è il solo flag di stato coinvolto dall'istruzione DAD. . Le istruzioni INX, DCX e DAD permettono il calcolo della ricerca tabellare". Sono usate anche per l'indirizzamento modificato da indice e la manipolazione dei dati in archivio.

## CMP r e CMP M

**CMP r** (Confrontare il registro)
(A) - (r)
Il contenuto del registro r è sottratto dall'accumulatore. L'accumulatore resta invariato. I flag vengono posizionati come risultato della sottrazione.
**Il flag Z è posizionato su 1 se (A) = (r).**
**Il flag CY è posizionato su 1 se (A) < (r).**

| 1 | 0 | 1 | 1 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

**CMP M** (Confrontare la memoria)
(A) − (H) (L)
Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L viene sottratto dall'accumulatore. L'accumulatore resta invariato.
I flag sono posizionati come risultato della sottrazione.
**Il flag Z è posizionato su 1 se (A) = ( (H) (L) )**
**Il flag CY è posizionato su 1 se (A) < ( (H) (L) ).**

| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

Citiamo il *"μCOM-8 Software Manual"*: "CMP r e CMP M sono usati per confrontare due *dati senza alterarli*. CMP r confronta i contenuti dell'accumulatore con uno dei singoli registri B,C,D,E,H e L; l'accumulatore A; o M, la posizione di memoria indirizzata dalla coppia di registri H,L.

L'istruzione non coinvolge nessuno dei registri di dati, ma coinvolge i quattro bit di flag Carry, Zero, Sign e Parity. Le istruzioni di confronto eseguono una sottrazione interna della sorgente S dall'accumulatore. I flag vengono settati sulla base di quello che sarebbe stato il risultato della sottrazione. Così Zero è settato se le quantità erano uguali, Sign è settato se il risultato era negativo (il bit più significativo è a livello logico 1), Parity è settato se il risultato è di parità, e Carry è settato se c'è un riporto negativo dal bit 7 (dati sorgente maggiori dei dati dell'accumulatore)."

"Così, in ogni caso:

Carry è settato se vi è un riporto negativo; altrimenti è resettato
Sign è settato uguale al MSB del risultato;
Zero è settato se il risultato è zero; altrimenti è resettato.
Parity è settato se la parità del risultato è even; altrimenti è resettato.

Le istruzioni di confronto sono usate nel modo migliore per i confronti aritmetici senza segno (numeri compresi nella fascia di valori da 0 a $255_{10}$), chiamati anche confronti logici o di caratteri. In questo caso, i risultati per i flag Zero e Carry possono essere interpretati nel modo seguente:

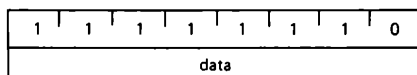
Risultato dell'operazione di confronto

| Flag di Zero | Flag di Riporto | Rapporto fra accumulatore e registro |
|---|---|---|
| 1 | X | Accumulatore = registro |
| X | 1 | Accumulatore < registro |
| 1 | 1 | Accumulatore ⩽ registro |
| 0 | 0 | Accumulatore > registro |
| X | 0 | Accumulatore ⩾ registro |

NOTA: X = non ha importanza

Così, le relazioni { = , <· ⩾ } possono essere provate usando una sola istruzione di salto, mentre { ⩽, > } ne richiedono due. Notate che se gli operandi vendono invertiti, > rimpiazza ⩽ e < rimpiazza ⩾."

**CPI data**

**CPI data** (Confrontare in modo immediato)
(A) — (byte 2)
Il contenuto del secondo byte dell'istruzione è sottratto dall'accumulatore. I flag sono posizionati dal risultato della sottrazione. Il flag Z è settato ad 1 se (A) = (byte 2). Il flag CY è settato ad 1 se (A) < (byte 2).

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P, CY, AC

L'istruzione CPI data è un'operazione immediata che confronta i contenuti dell'accumulatore con la quantità di 8 bit nel secondo byte dell'istruzione. L'istruzione coinvolge tutti e cinque i flag, ma solo quattro di questi producono risultati utili. I flag vengono settati o azzerati sulla base di quello che avrebbe dovuto essere il risultato della sottrazione. *I contenuti dell'accumulatore restano invariati*. Per ulteriori particolari, vedere la discussione precedente sull'istruzione CMP r.

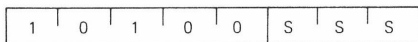
Si può dedurre che le istruzioni CMP r e CPI data sono operazioni logiche e non aritmetiche. Dal momento che viene eseguita un'operazione aritmetica - la sottrazione - le includeremo nel gruppo delle operazioni aritmetiche. Lo scopo delle istruzioni di confronto è produrre decisioni che vengono riflesse negli stati logici dei bit di flag.

## GRUPPO LOGICO

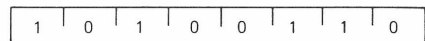
Questo gruppo di istruzioni esegue operazioni logiche, cioè Booleane, su dati nei registri e nella memoria e sui flag di condizione. Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Sign, Parity, Carry, Auxiliary Carry e Carry secondo le regole standard.

### ANA r e ANA M

**ANA r** (AND del registro)
$(A) \leftarrow (A) \wedge (r)$
Il contenuto del registro r è addizionato in maniera logica al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.
**I flag CY e AC vengono azzerati.**

| 1 | 0 | 1 | 0 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento registro
Flag: Z,S,P,CY,AC

**ANA M** (AND della memoria)
$(A) \leftarrow (A) \wedge ((H)\ (L))$
I contenuti della locazione di memoria il cui indirizzo è contenuto nei registri H ed L è addizionato in maniera logica al contenuto dell'accumulatore.
Il risultato si trova nell'accumulatore.
**I flag CY e AC vengono azzerati.**

| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione ANA r esegue un *AND logico bit a bit parallelo* dei contenuti dell'accumulatore e dei contenuti del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri generali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti della posizione di memoria indirizzati dalla coppia di registri H,L. Per esempio, l'operazione ANA B esegue un'operazione AND logica bit a bit con i contenuti del registro B e dell'accumulatore. Il caso particolare di

**ANA A**

azzera il flag di Carry e fa sì che il flag di Zero venga settato se il risultato è zero, azzerato se il risultato non è zero. Tutti i flag sono coinvolti dall'istruzione ANA r. Dal momento che $A \cdot A = A$, i dati nell'accumulatore non sono cambiati. Questo è un "trucco" per azzerare il flag di riporto o semplicemente per testare se l'accumulatore ha valore zero.

## ANI data

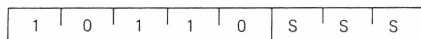
**ANI data** (AND in modo immediato)
(A) ← (A) ∧ (byte 2)
Il contenuto del secondo byte dell'istruzione è addizionato in maniera logica ai contenuti dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

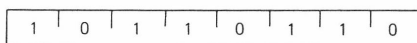
Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ANI data esegue un AND logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag vengono coinvolti da questa istruzione.

## ORA r e ORA M

**ORA r** (OR del registro)
(A) ← (A) V (r)
Viene eseguita un'operazione di OR inclusivo con il contenuto del registro r e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.
**I flag CY e AC vengono azzerati.**

| 1 | 0 | 1 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

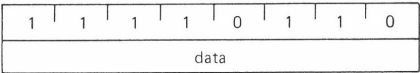
**ORA M** (OR della memoria)
(A) ← (A) V ( (H) (L) )
Viene eseguita un'operazione di OR inclusivo con il contenuto della locazione di memoria, il cui indirizzo è contenuto nei registri H ed L, e il contenuto dell'accumulatore.
Il risultato si trova nell'accumulatore.
**I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione ORA r esegue un *OR logico bit a bit parallelo* dei contenuti dell'accumulatore e dei contenuti del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri universali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti della locazione di memoria indirizzata dalla coppia di registri H,L.
Il comando,

**ORA A**

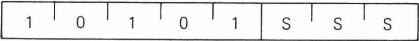
Che ha il codice di istruzione ottale 267, è un sistema conveniente per azzerare il flag di riporto senza coinvolgere nient'altro.
Sia ORA r che l'istruzione a due byte relativa, ORI data, azzerano il flag di riporto e fanno sì che il flag di Zero venga settato se il risultato è zero, azzerato se il risultato non è zero.

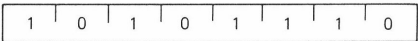
## ORI data

**ORI data** ((OR in modo immediato)
(A) ← (A) ∀ (byte 2)
Viene eseguita un'operazione di OR inclusivo del contenuto del secondo byte dell'istruzione con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ORI data esegue un OR logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag sono coinvolti da questa istruzione.

## XRA r e XRA M

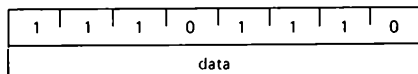
**XRA r** (OR esclusivo del registro)
(A) ← (A) ∀ (r)
Viene eseguita un'operazione di OR esclusivo del contenuto del registro r con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

**XRA M** (OR esclusivo della memoria)
(A) ← (A) ∀ ( (H) (L) )
Viene eseguita un'operazione di OR esclusivo del contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.
**I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione XRA r esegue un *OR esclusivo logico bit a bit* dei contenuti dell'accumulatore e del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri universali B, C, D, E, H ed L; l'accumulatore A; o M, la locazione di memoria indirizzata dalla coppia di registri H,L. Tutti i flag sono coinvolti da questa istruzione.

## XRI data

L'istruzione XRI data esegue un OR esclusivo logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag sono coinvolti dall'istruzione.

**XRI data** (OR esclusivo in modo immediato)
$(A) \leftarrow (A) \veebar (\text{byte } 2)$
Viene eseguita un'operazione di OR esclusivo del contenuto del secondo byte dell'istruzione con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

Citiamo il *μCOM-8 Software Manual* della NEC Microcomputers, Inc.: "L'istruzione logica suddetta verrà usata per implementare una tecnica di programmazione conosciuta come *"masking"* (mascheratura). Si tratta di una tecnica per mezzo della quale i bit di un operando vengono selettivamente modificati per essere usati in una successiva operazione. Vi sono tre tipi generali di "masking":

- Azzerare tutti i bit sui quali non si opera
- Settare tutti i bit sui quali non si opera (usati raramente)
- Lasciare inalterati tutti i bit sui quali non si opera.

I primi due vengono chiamati *"masking esclusivo"*, e il terzo *"masking inclusivo"*. Ad esempio, supponiamo che l'accumulatore contenga il seguente valore,

bit: 7 6 5 4 3 2 1 0
1 1 0 1 $^{1}_{0}$ 1 1 0 Contenuti dell'accumulatore

Per provare se il bit 3 è zero o uno e simultaneamente azzerare gli altri bit, l'accumulatore è mascherato con 00001000. Usando l'istruzione,

ANI
010

l'accumulatore conterrà zeri con il flag di Zero settato se il bit di 3 era stato uno zero, e conterrà 010, in codice ottale, con il flag di Zero azzerato se il bit 3 era stato uno".

"Allo scopo di posizionare il bit 3 su uno e lasciare gli altri bit inalterati, la stessa configurazione di bit viene usata, assieme all'istruzione,

ORI
010

Il risultato in questo caso è 11011110 nell'accumulatore".

"Allo scopo di posizionare il bit 3 su zero e lasciare gli altri bit inalterati, l'accumulatore è addizionato con 1111011, il complemento della maschera del primo esempio. Con l'istruzione.

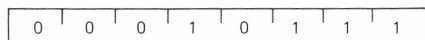
ANI
367

il risultato dell'accumulatore è 11010110. Queste sono le operazioni di manipolazione dei bit usate più comunemente, dato che la mascheratura viene eseguita in una sola istruzione.

Ne sono possibili molte altre, ma spesso richiedono più di un'istruzione per l'implementazione."

**ISTRUZIONI DI ROTAZIONE**

Tutte le istruzioni di rotazione 8080A sono riassunte nello schema seguente:

## RAL e RAR

L'istruzione RAL, o "Ruotare l'Accumulatore a Sinistra", fa sì che l'accumulatore faccia ruotare tutti i bit di una posizione verso sinistra attraverso il bit di riporto, cioè una rotazione a 9 bit: Il bit 7 si trasferisce al flag di riporto, il bit di riporto si trasferisce al bit 0, il bit 0 si trasferisce al bit 1, il bit 1 si trasferisce al bit 2, e così via, come mostra la pagina precedente.

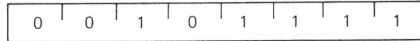
L'istruzione RAR, o "Ruotare l'Accumulatore a Destra", fa sì che l'accumulatore faccia ruotare tutti i bit di una posizione verso destra attraverso il bit di riporto, cioè una rotazione a 9 bit. Il bit 0 si trasferisce al flag di riporto, il bit di riporto si trasferisce al bit 7, il bit 7 si trasferisce al bit 6, e così via, come mostra la pagina precedente.

## RLC e RRC

L'istruzione RLC, o "Ruotare a Sinistra Circolarmente", fa ruotare l'accumulatore di un bit verso sinistra e nel flag di riporto, come mostra lo schema di pagina precedente.

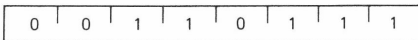
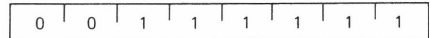
L'istruzione RRC, o "Ruotare a Destra Circolarmente", fa ruotare l'accumulatore di un bit verso destra e nel flag di riporto, come mostra lo schema di pagina precedente.

In entrambe queste istruzioni, l'informazione originale che appare sul flag di riporto, viene perduta.

## CMA

L'istruzione CMA complementa i contenuti dell'accumulatore senza coinvolgere nessuno dei bit di flag. Per esempio, se l'accumulatore contenesse 11010001, l'istruzione

CMA

lo convertirebbe in 00101110. Viene complementato ogni singolo bit.

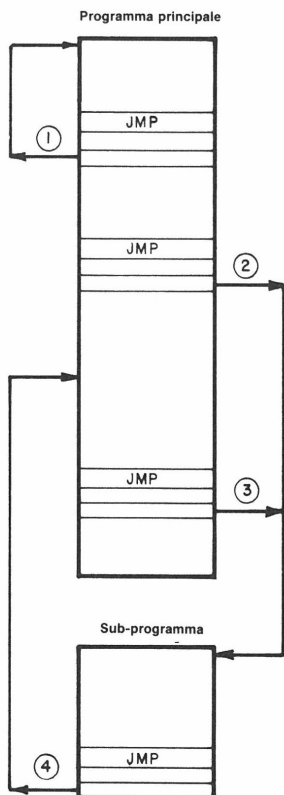
## STC e CMC

L'istruzione STC setta il flag di riporto sul livello logico 1; l'istruzione CMC complementa il flag di riporto. Non è coinvolto nessun altro bit di flag.

## GRUPPO DI SALTO

Questo gruppo di istruzioni altera il normale flusso sequenziale di un programma. *I flag di condizione non sono coinvolti da nessuna istruzione di questo gruppo.* I due tipi di istruzioni di salto sono incondizionato e condizionato. I trasferimenti incondizionati eseguono semplicemente l'operazione specificata sul registro PC, il contatore di programma. I trasferimenti condizionati esaminano lo stato di uno dei quattro flag di processo - Zero, Sign, Parity, o Carry - per determinare se l'operazione di salto specifica deve essere eseguita. Le condizioni che possono essere specificate sono le seguenti:

| CONDIZIONE | CCC |
|---|---|
| NZ - non zero (Z=0) | 000 |
| Z - zero (Z =1) | 001 |
| NC - nessun riporto (CY = 0) | 010 |
| C - riporto (CY = 1) | 011 |
| PO - parità dispari (P = 0) | 100 |
| PE - parità pari (P = 1) | 101 |
| P - più (S = 0) | 110 |
| M - meno (S = 1) | 111 |

NOTA: CCC è il codice a tre bit per la condizione dei flag

## JMP addr

**JMP addr** (Salto)
(PC) ← (byte 3) (byte 2)
Il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 2 e 3 di questa istruzione.

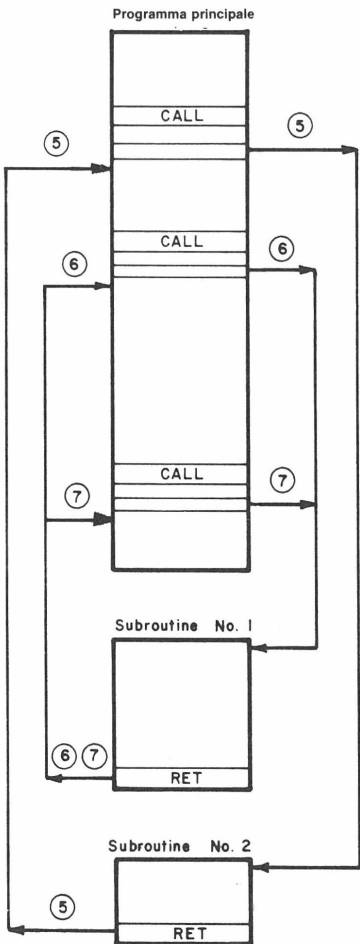
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

Il *contatore di programma (program counter)* è il registro a 16 bit nel microprocessore 8080A che contiene l'indirizzo di memoria del byte di istruzione successivo che deve essere eseguito in un programma. L'istruzione JMP addr è semplicemente un'istruzione di trasferimento di byte, nella quale il secondo e il terzo byte istruzioni vengono trasferiti direttamente

al contatore di programma. Non sono incluse operazioni aritmetiche nè logiche, e non è coinvolto nessun bit di flag. L'istruzione JMP è un'istruzione a tre byte che contiene l'indirizzo di memoria a 16 bit al quale viene trasferito il controllo di programma. Potete saltare avanti o indietro a tutte le 65.536 possibili locazioni di memoria. Il microprocessore non ricorda il punto dal quale è saltato, in netto contrasto con il comportamento delle istruzioni CALL e RET descritte più avanti.

Il comportamento dell'istruzione JMP può essere compreso con l'aiuto dello schema mostrato precedentemente. La prima istruzione JMP, (1), è un salto indietro che crea un loop. JMP (2) e JMP (3) trasferiscono il controllo di programma al sottoprogramma. L'uscita del sottoprogramma è designata dall'istruzione JMP (4).

## CALL addr e RET

**CALL addr** (Richiamo)

( (SP) − 1) ← (PCH)
( (SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

Gli otto bit di ordine più elevato dell'indirizzo dell'istruzione successiva sono posti nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP. Gli otto bit di ordine inferiore dell'indirizzo dell'istruzione successiva sono posti nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di 2. Il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso.

Cicli: 5
Stati: 17
Indirizzamento: immediato/reg. indiretto
Flag: nessuno

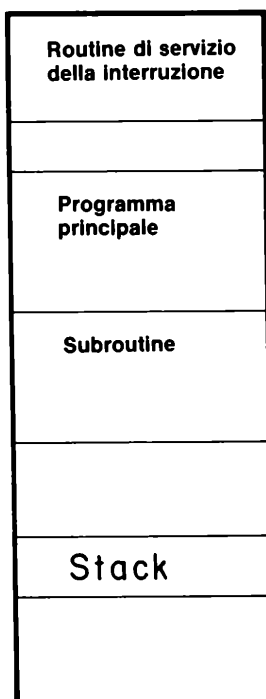
**RET** (Ritorno)

(PCL) ← ( (SP) );
(PCH) ← ( (SP) + 1);
(SP) ← (SP) + 2;

Il contenuto della locazione di memoria il cui indirizzo è specificato nel registro SP, è posto negli otto bit di ordine inferiore del registro PC. Il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP, è spostato agli otto bit di ordine più elevato del registro PC. Il contenuto del registro SP è incrementato di 2.

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: nessuno

Può darsi che spesso vogliate "saltar fuori" da un programma principale, ma ritornarvi più tardi. Per farlo, dovete non solo conoscere la vostra nuova destinazione, ma dovete anche ricordare in qualche modo la vostra locazione originale. A questo scopo, avete due tipi di istruzioni, richiamo di subroutine e rientro dalla subroutine. Qui parleremo delle istruzioni incondizionate CALL addr e RET. Citiamo il Manuale della NEC Microcomputers: "L'istruzione CALL trasferisce il controllo ad una subroutine. L'istruzione CALL addr salva il contatore di programma incrementato nello *stack* e pone l'indirizzo nel contatore di programma. Lo stack è un blocco di memoria lettura/scrittura indirizzato da uno speciale registro a 16 bit conosciuto come "stack pointer" che può essere caricato dall'utente (LXI SP, data 16). Lo stack opera come una memoria "last-in-first-out" (LIFO), con il registro stack pointer che indirizza il dato più recente posto nello stack. L'istruzione di rientro fa sì che il puntatore dello stack venga posto nel contatore di programma. Così un'istruzione di richiamo (CALL) trasferisce il controllo di programma dal programma principale nella subroutine e un'istruzione RET ritrasferisce il controllo al programma principale."

La posizione dello *stack* è di solito agli indirizzi più alti nella memoria disponibile di un microcomputer basato sull'8080A. Nel diagramma seguente, lo stack è ad una certa distanza dal programma principale e dalle subroutines.

**JNZ, JZ, JNC, JC, JPO, JPE, JP,** e **JM addr**

**Jcondition addr** (Salto condizionato)
Se (CCC),
(PC) ← (byte 3) (byte 2)
Se la condizione specificata è vera, il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso; altrimenti, il controllo continua sequenzialmente.

| 1 | 1 | C | C | C | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

In un'istruzione di salto condizionato, se la condizione è soddisfatta, il secondo e il terzo byte dell'istruzione vengono trasferiti al contatore di programma e si verifica un salto. Se la condizione non è soddisfatta, non avviene nessun cambiamento nel contatore di programma; il controllo di programma passa all'istruzione che segue immediatamente il salto.

Le varie condizioni possono essere riassunte come segue:

NZ: Il risultato a 8 bit nell'operazione aritmetica o logica immediatamente precedente Non è uguale a Zero, cioè il flag Zero è stato azzerato.

Z: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente è uguale a Zero, cioè il flag di Zero è stato settato.

NC: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente non produce nessun riporto (No Carry) dal bit più significativo; o, il flag di riporto è stato azzerato.

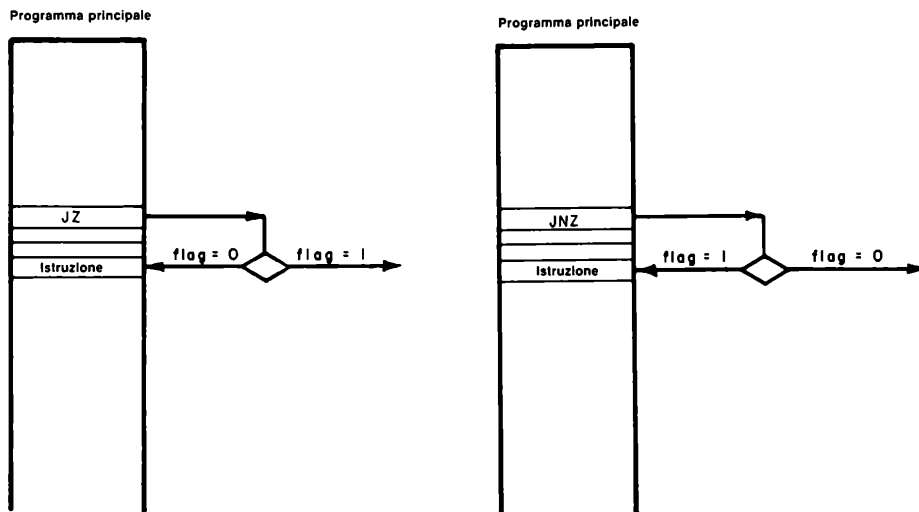
C: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un riporto (Carry) dal bit più significativo; o, il flag di riporto è settato.

PO: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente ha una Parità che è "Odd", cioè il flag di parità è stato azzerato.

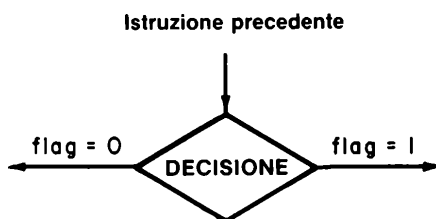
PE: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente ha una Parità che è "Even", cioè il flag di parità è stato settato.

P: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un MSB che ha un segno Più, cioè il flag di segno è stato azzerato.

M: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un MSB che ha un segno Meno, cioè il flag di segno è stato settato.

Il valore di CCC che corrisponde ad ognuna delle condizioni è stato mostrato parecchie pagine indietro. Il comportamento di due delle istruzioni condizionate, JNZ e JZ, può essere compreso con l'aiuto del diagramma seguente:

Nell'istruzione JNZ, il salto avviene solo se il risultato a 8 bit di un'operazione aritmetica o logica Non è Zero. Il simbolo di scelta logica,

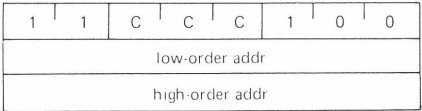
**Istruzione precedente**

flag = 0 DECISIONE flag = 1

che viene usato nei flow-chart, indica che ciò che succederà poi dipende dallo stato del flag di Zero. Per JNZ, avviene un salto se il flag di Zero è azzerato, cioè a livello logico 0. Per JZ, avviene un salto se il risultato a 8 bit è uguale a zero; in tal caso, il flag di Zero è a livello logico 1.

È possibile confondersi a proposito delle condizioni NZ e Z. Notate che NZ e Z si riferiscono al risultato a 8 bit di un'operazione, non allo stato logico del flag di Zero. NZ significa che il risultato a 8 bit di un'operazione non è zero; Z significa che il risultato a 8 bit di un'operazione è zero (sebbene il flag di Zero sia a livello logico 1). Nel discutere l'argomento, abbiamo tentato di dimostrare che una condizione può essere vista come il risultato a 8 bit di un'operazione aritmetico/logica (NZ,Z,NC,C,PO,PE,P,M) o come lo stato logico dei singoli flag che prova il risultato di un'operazione aritmetico/logica.

Preferiamo usare il risultato a 8 bit di un'operazione ALU, compresi i simboli letterali NZ,Z,NC, ecc. Speriamo di non avervi messo in confusione.

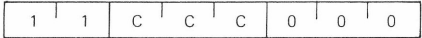
### CNZ, CZ, CNC, CC, CPO, CPE, CP e CM addr

**Condition addr** (Richiamo condizionato)
Se (CCC),
( (SP) − 1) ← (PCH)
( (SP) − 2) ← (PCL)
(SP) ← ((SP) − 2
(PC) ← (byte 3) (byte 2)
Se la condizione specificata è vera, vengono eseguite le azioni specificate nell'istruzione di CALL (vedi sopra); altrimenti, il controllo continua sequenzialmente.

Cicli: 3/5
Stati: 11/17
Indirizzamento: immediato/reg.indiretto
Flag: nessuno

In un'istruzione di richiamo *(call)* condizionato, se la condizione è soddisfatta, viene richiamata la subroutine della locazione di memoria data nel secondo e nel terzo byte di istruzioni. I contenuti del contatore di programma sono posti sullo stack, in modo che un'istruzione di rientro potrà rimandare il controllo di programma all'istruzione che segue immediatamente l'istruzione di salto condizionato.

Se la condizione non è soddisfatta, l'esecuzione del programma passa all'istruzione che segue immediatamente l'istruzione di richiamo condizionato.

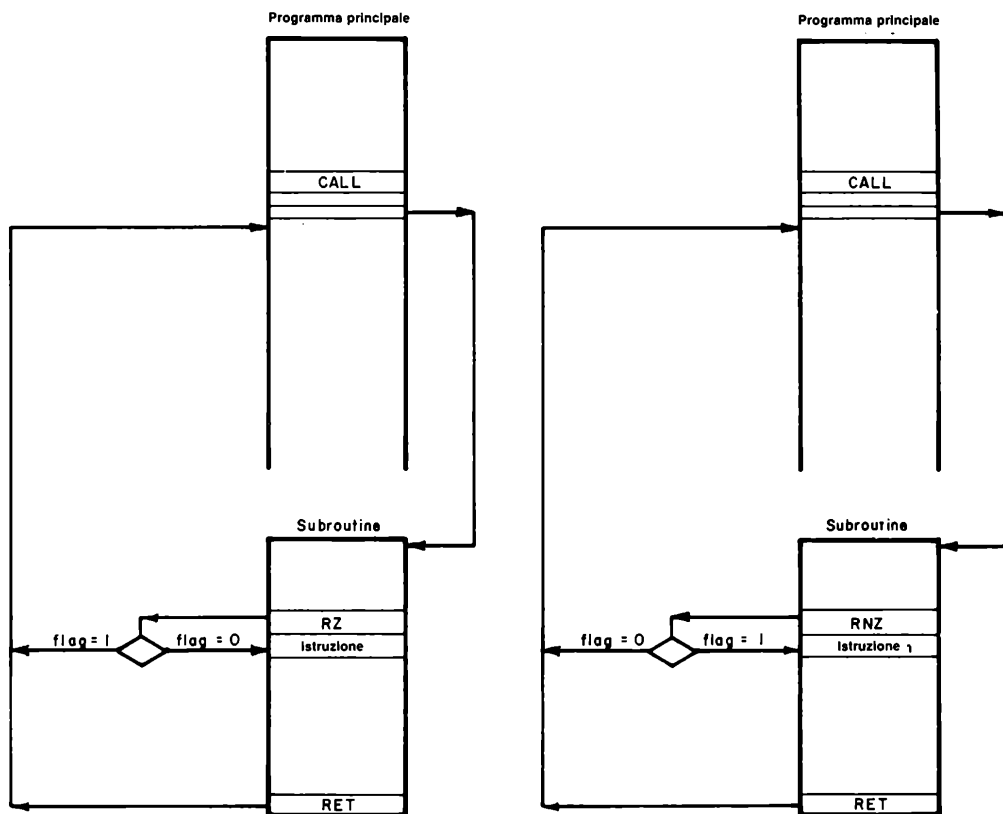
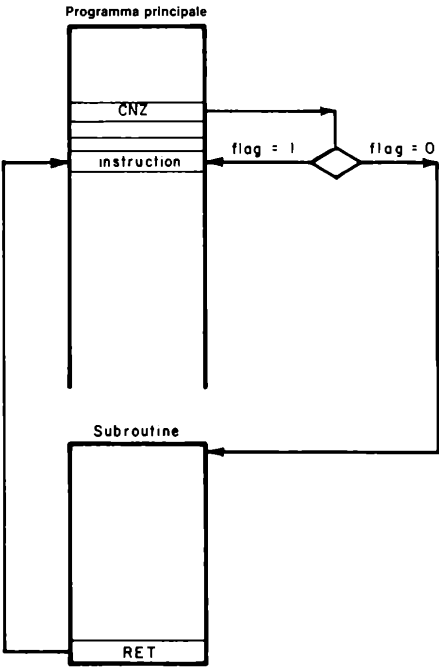
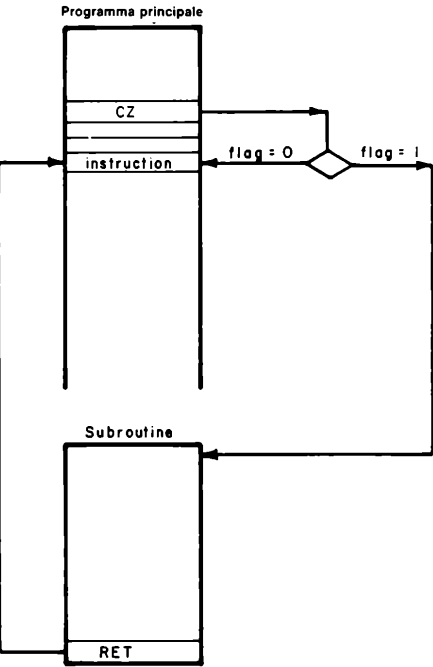
### RNZ, RZ, RNC, RC, RPO, RPE, RP, e RM

**Rcondition** (Rientro condizionato)
Se (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
Se la condizione specificata è vera, vengono eseguite le azioni specificate nell'istruzione RET (vedi sopra); altrimenti, il controllo continua sequenzialmente.

Cicli: 1/3
Stati: 5/11
Indirizzamento: reg. indiretto
Flag: nessuno

In un'istruzione di rientro condizionato, se la condizione è soddisfatta, avviene un rientro dalla subroutine; i contenuti del contatore di programma sullo stack vengono trasferiti al contatore di programma e l'esecuzione riprende all'istruzione immediatamente dopo l'istruzione di richiamo della subroutine.

Se la condizione non è soddisfatta, l'esecuzione del programma passa all'istruzione che segue immediatamente l'istruzione di rientro condizionato.

Le istruzioni condizionate, CZ, CNZ, RZ, e RNZ sono illustrate schematicamente nel diagramma che segue. Ricordate, Z significa che il flag di Zero dev'essere a livello logico 1 perchè avvenga un richiamo o un rientro; altrimenti, il controllo di programma passa all'istruzione successiva. NZ significa che il flag di Zero deve essere a livello logico 0 perchè avvenga un richiamo o un rientro; altrimenti, il controllo di programma passa all'istruzione successiva.

Programma principale
CZ
instruction
flag = 0
flag = 1
Subroutine
RET
Programma principale
CNZ
instruction
flag = 1
flag = 0
Subroutine
RET

**RSTn**

**RST n** (Ripristino)
( (SP) − 1) ← (PCH)
( (SP) − 2) ← (PCL)
(SP) ← (SP) − 2
PC) ← 8 * (NNN)
Gli otto bit di ordine più elevato dell'indirizzo dell'istruzione successiva vengono spostati nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP. Gli otto bit di ordine inferiore dell'indirizzo dell'istruzione successiva vengono spostati nella locazione di memoria il cui indirzzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di due. Il controllo è trasferito all'istruzione il cui indirizzo è otto volte il contenuto di NNN.

| 1 | 1 | N | N | N | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3

Stati: 11

Indirizzamento: reg. indiretto

Flag: nessuno

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N | N | N | 0 | 0 | 0 |

Contatore di programma dopo il ripristino

Citiamo il "*μCOM-8 Software Manual*": "Le istruzioni EI (Abilitare l'interruzione) e DI (Disabilitare l'interruzione) effettuano il controllo sull'accettazione di una richiesta di interruzione. Una volta che questo controllo è stabilito, il problema seguente da risolvere è come il dispositivo esterno indica al processore dove è posizionata la routine d'interruzione desiderata. L'8080A procede a questa identificazione permettendo al dispositivo di fornire un'istruzione quando viene riconosciuta l'interruzione. Benchè tutte le istruzioni 8080A possano essere specificate, solo due hanno valore pratico: un'istruzione di richiamo, CALL, e un'istruzione di ripristino, RST . . . . . Un'istruzione RST è un tipo specializzato di CALL. L'istruzione RST è un richiamo ad una delle otto locazioni in memoria specificata da un'espressione intera nella fascia di valori da 0 a 7 in codice ottale, indicata da N. Segue un elenco delle locazioni specificate dagli interi da 0 a 7.

| Valore di N | Locazione richiamata |
|---|---|
| 0 | HI = 000 e LO = 000 |
| 1 | HI = 000 e LO = 010 |
| 2 | HI = 000 e LO = 020 |
| 3 | HI = 000 e LO = 030 |
| 4 | HI = 000 e LO = 040 |
| 5 | HI = 000 e LO = 050 |
| 6 | HI = 000 e LO = 060 |
| 7 | HI = 000 e LO = 070 |

Un'istruzione RST fa sì che il contatore di programma incrementato venga

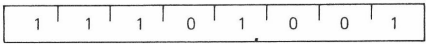
inserito nello stack esattamente come accade con un'istruzione CALL. Carica poi il contatore di programma con HI = 000 e LO = ONO, dove N va da 0 a 7. RST 4 fa sì che il contatore di programma venga inserito nello stack e che HI = 000 e 000 e LO = 040 entrino nel contatore di programma."

"L'esecuzione del programma continua poi dal punto di ripristino. Se la routine di servizio dispositivo richiede più di otto byte per funzionare (come accade nella maggior parte dei casi), l'istruzione posta nel punto di ripristino deve determinare un salto alla subroutine di servizio interruzione. Dato che RST è un richiamo di subroutine specializzato, la subroutine di servizio interruzione *deve terminare con un'istruzione di rientro,* per ridare il controllo al programma interrotto riprendendo l'indirizzo di rientro."

"Dato che l'8080A ha soltanto otto istruzioni RST, qualunque livello aggiuntivo di interruzione deve essere implementato usando istruzioni CALL. Ciò significa che un'istruzione CALL addr deve essere fornita dal dispositivo d'interruzione, che è più difficile da implementare in hardware perchè CALL è un'istruzione a 3 byte. Comunque, una volta implementato, un richiamo diretto ad un routine è leggermente più veloce di un ripristino e di una successiva operazione di salto. Benchè questo non sia il fattore più importante, questa differenza in velocità di risposta dovrebbe essere tenuta in considerazione nel determinare come implementare le routine di servizio interruzione. Il primo beneficio che si ottiene usando la CALL è che un sistema di vettorizzazione ad "n" vie è fatto da hardware, eliminado il bisogno di software nella memoria di ordine inferiore (per l'elaborazione RST). Così quelle locazioni di memoria diventano libere per i programmi d'utente.

**PCHL**

**PCHL** (Saltare ad H ed L indirettamente - spostare H ed L nel PC)
(PCH) ← (H)
(PCL) ← (L)
Il contenuto del registro H è spostato negli otto bit di ordine più elevato del registro PC. Il contenuto del registro L è spostato negli otto bit di ordine inferiore del registro PC.

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

L'istruzione PCHL fa sì che il contatore di programma venga caricato con i contenuti della coppia di registri HL. L'esecuzione del programma continua poi al punto designato dal contenuto di HL. In effetti, questa è un'istruzione di salto, ma, dal momento che sulla coppia di registri HL si può operare aritmeticamente, si possono implementare molti salti calcolati. La sequenza di istruzioni,

LXI H
<B2>
<B3>
PCHL

è identica in funzione a

JMP
<B2>
<B3>

## GRUPPO STACK, I/O, CONTROLLO MACCHINA

Questo gruppo di istruzioni esegue I/0, manipola lo stack, e altera i flag di controllo interni. Salvo diversa indicazione, *i flag di condizione non vengono coinvolti da nessuna istruzione di questo gruppo.*

### PUSH rp e POP rp

**PUSH rp** (Inserire nello stack)

( (SP) − 1) ← (rh)
( (SP) − 2) ← (rl)
(SP) ← (SP) − 2

Il contenuto del registro di ordine più elevato della coppia di registri rp è spostato nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP. Il contenuto del registro di ordine inferiore della coppia di registri rp è spostato nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di 2. **Nota: La coppia di registri rp=SP può non essere specificata.**

| 1 | 1 | R | P | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3
Stati: 11
Indirizzamento: reg. indiretto
Flag: nessuno

**POP rp** (estrarre dallo stack)

(rl) ← ( (SP) )
(rh) ← ( (SP) + 1)
(SP) ← (SP) + 2

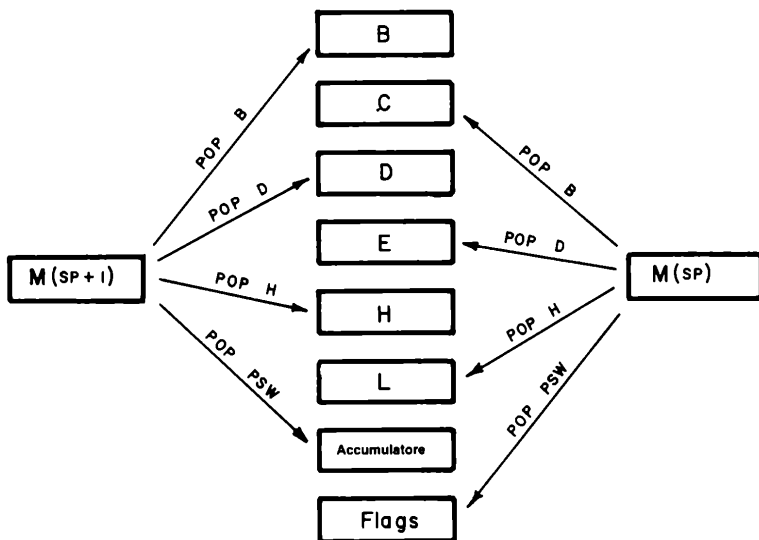
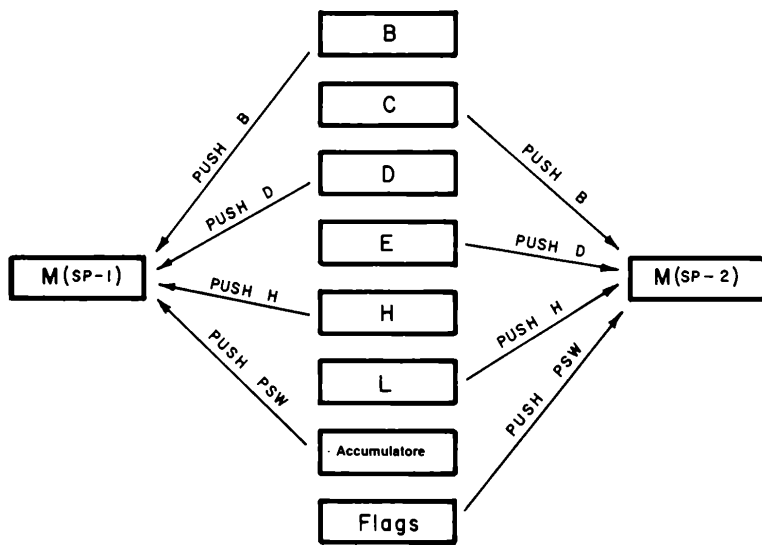
Il contenuto della locazione di memoria, il cui indirizzo è specificato dal contenuto del registro SP, è spostato nel registro di ordine inferiore della coppia di registri rp. Il contenuto della locazione di memoria, il cui indirizzo è uno più del contenuto del registro SP, è spostato nel registro di ordine più elevato della coppia di registri rp. Il contenuto del registro SP è incrementato di 2. **Nota: La coppia di registri rp=SP può non essere specificata.**

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: nessuno

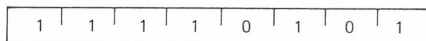
Citiamo il *"μCOM-8 Software Manual"*: "Due speciali istruzioni permettono al programmatore di mantenere e di reinserire in memoria i registri usando lo stack, PUSH e POP. PUSH rp fa sì che la coppia di registri specificata da RP venga posta nello stack. Lo stack è una parte speciale della memoria di lettura/scrittura designata dall'utente e usata come una memoria last-in-first-out (LIFO) per mezzo dello stack pointer a 16 bit. Un'operazione PUSH fa sì che lo stack pointer si decrementi di uno e memorizzi il registro più significativo (il registro HI) nella nuova locazione di memoria specificata dallo stack pointer. Lo stack pointer viene poi decrementato di nuovo e il registro meno significativo (il registro LO) viene poi memorizzato a quell'indirizzo. Per un'operazione POP, i dati nella locazione di memoria indirizzata dallo stack pointer vengono spostati nel registro meno significativo (il registro LO, che può essere C, E, o L); lo stack pointer è incrementato e i dati nella nuova locazione di memoria vengono caricati nel registro più significativo (il registro HI, che può essere B, D o H). Lo stack pointer viene poi incrementato di nuovo."

"Sia per le operazioni PUSH che POP, la coppia di registri, RP, può essere una delle tre coppie di registri BC, DE, o HL (identificate come B, D, e H, rispettivamente) o i contenuti del registro dei Flag e accumulatore, indicato da PSW (che sta per "program status word").

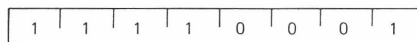
Le istruzioni PUSH e POP sono rappresentate schematicamente nella figura di pagina seguente. In questo diagramma, SP è la locazione originaria dello stack pointer, prima delle istruzioni PUSH o POP.

B
C
D
E
H
L
Accumulatore
Flags
M (SP-1)
M (SP-2)
PUSH B
PUSH D
PUSH H
PUSH PSW
B
C
D
E
H
L
Accumulatore
Flags
M (SP+1)
M (SP)
POP B
POP D
POP H
POP PSW

## PSH psw e POP psw

**PUSH psw** (Inserire la parola di stato)
$((SP)-1) \leftarrow (A)$
$((SP)-2)_0 \leftarrow (CY)$, $((SP)-2)_1 \leftarrow 1$
$((SP)-2)_2 \leftarrow (P)$, $((SP)-2)_3 \leftarrow 0$
$((SP)-2)_4 \leftarrow (AC)$, $((SP)-2)_5 \leftarrow 0$
$((SP)-2)_6 \leftarrow (Z)$, $((SP)-2)_7 \leftarrow (S)$
$(SP) \leftarrow (SP)-2$
Il contenuto del registro A è spostato nella locazione di memoria il cui indirizzo è uno meno del registro SP. I contenuti dei flag vengono assemblati nella parola di stato e la parola è spostata nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di due.

| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3
Stati: 11
Indirizzamento: reg. indiretto
Flag: nessuno

**POP psw** (Estrarre la parola di stato)
$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
$(A) \leftarrow ((SP)+1)$
$(SP) \leftarrow (SP)+2$
Il contenuto della locazione di memoria il cui indirizzo è specificato dal contenuto del registro SP viene usato per reinserire in memoria i flag di condizione. Il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP, è spostato nel registro A. Il contenuto del registro SP è incrementato di 2.

| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

### PAROLA DEI FLAG

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| S | Z | 0 | AC | 0 | P | 1 | CY |

Le lettere, PSW, stanno per *"program status word";* nel PSW vi sono i contenuti dell'accumulatore e dei cinque flag stato. Vi rimandiamo alla descrizione delle istruzioni PUSH rp e POP rp delle pagine precedenti. Il registro dei flag, F, è considerato il registro più significativo e l'accumulatore, A, è considerato il registro meno significativo. Il program status word è importante perchè mantiene lo stato macchina attuale così come è stato determinato dai cinque bit di flag. Quando PSW viene reinserito in memoria, le operazioni macchina possono riprendere dallo stato corretto, a prescindere da come la subroutine che ha interrotto coinvolge i flag.

Nel circuito integrato $\mu$COM-8, che è funzionalmente identico al microprocessore 8080A, c'è un flag di stato in più, SUB. Nel registro dei flag, SUB occupa la posizione bit $D_5$. Inoltre, la posizione bit $D_3$ è a livello logico 1 e non a livello logico 0 (che è il caso del chip 8080A). Pensiamo che il flag SUB sia una caratteristica utile dei microprocessori tipo 8080A, e speriamo che venga incorporato nelle versioni future del chip dai fabbricanti, quali Texas Instruments, National Semiconductor, Intel, Siemens, ecc.

Un esempio di come opera lo stack viene dato nella figura di pagina seguente. La sezione di programma usata è,

| | *Subroutine* |
|---|---|
| CALL | PUSH B |
| <B2> | PUSH D |
| <B3> | PUSH H |
| | PUSH PSW |

Originariamente lo stack pointer era posizionato a HI = 003 e LO = 303.
Dopo l'istruzione CALL, i due contatori di programma vengono inseriti sullo stack e lo stack pointer si sposta su HI = 003 e LO = 301. Notate che il byte del contatore di programma HI va sullo stack per primo ma se ne allontana per ultimo. Una successione di quattro istruzioni PUSH carica lo stack con i contenuti dei sei registri universali, dell'accumulatore, e del registro dei flag. Dopo tutto ciò, la locazione dello stack pointer (SP) è HI = 003 e LO = 271, la posizione occupata in alto sullo stack.

Una volta che la subroutine è stata eseguita, vi è il problema di spostare i contenuti dello stack e di rimetterli nel microprocessore 8080A. La sezione di programma, posizionata alla fine della subroutine, che assolve questo compito, è

```
POP PSW
POP H
POP D
POP B
RET
```

In ogni caso, il byte LO si allontana dallo stack per primo. Ricordate che nelle istruzioni a tre byte, il byte LO è sempre il secondo byte dell'istruzione. Quindi, il chip 8080A è ben preciso nella sua gestione di parole di 16 bit. Dopo che i contenuti dello stack sono stati estratti, lo stack pointer riprende la sua locazione originale ad HI = 003 e LO = 303.

I registri possono essere inseriti e prelevati in qualunque ordine. Comunque, il contatore di programma è quasi sempre inserito per primo e prelevato per ultimo. Dovete stare attenti a prelevare i registri nell'ordine inverso a quello nel quale li avete inseriti. Ad esempio, con la configurazione dello stack mostrata nella pagina seguente, se eseguiste la seguente sezione di programma alla fine della subroutine,

```
POP PSW
POP B
POP H
POP D
RET
```

avreste dei problemi nell'eseguire il programma principale. I contenuti originari del registro non ritornerebbero nelle loro locazioni originarie. Il chip tenterebbe di eseguire il programma, ma vi sarebbero poche possibilità di ottenere un risultato utile.

Se non avete bisogno di inserire il registro sullo stack durante un richiamo della subroutine, non fatelo.Memorizzate solo quell'informazione di cui il chip 8080A ha bisogno quando riprende il programma principale.

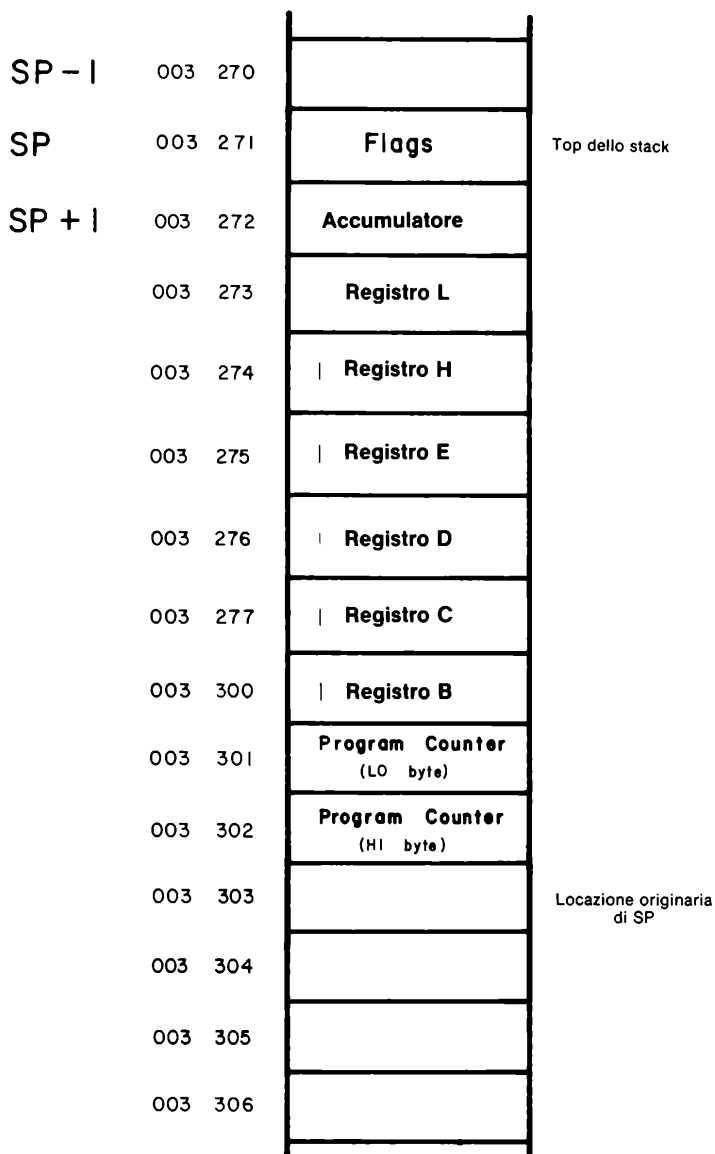
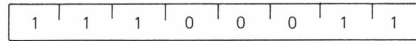
| | Indirizzo | Contenuto | |
|---|---|---|---|
| SP − 1 | 003 270 | | |
| SP | 003 271 | Flags | Top dello stack |
| SP + 1 | 003 272 | Accumulatore | |
| | 003 273 | Registro L | |
| | 003 274 | Registro H | |
| | 003 275 | Registro E | |
| | 003 276 | Registro D | |
| | 003 277 | Registro C | |
| | 003 300 | Registro B | |
| | 003 301 | Program Counter (LO byte) | |
| | 003 302 | Program Counter (HI byte) | |
| | 003 303 | | Locazione originaria di SP |
| | 003 304 | | |
| | 003 305 | | |
| | 003 306 | | |

**Lo stack**

*Figura 18-3.* Lo "stack".

## XTHL

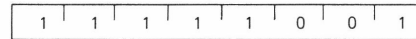
**XTHL** (Scambiare la parte superiore dello stack con H ed L)
(L) $\leftrightarrow$ ( (SP) )
(H) $\leftrightarrow$ ( (SP) + 1)
Il contenuto del registro L è scambiato con il contenuto della locazione di memoria il cui indirizzo è specificato dal contenuto del registro SP. Il contenuto del registro H è scambiato con il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP.

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 5
Stati: 18
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione XTHL viene usata per scambiare i contenuti della coppia di registri HL con la coppia superiore di elementi sullo stack. I contenuti della locazione superiore, quelli indirizzati dallo stack pointer SP, vengono scambiati con i contenuti del registro L. Lo stack pointer è incrementato, e i contenuti di memoria indirizzati da questo nuovo valore di SP, vengono scambiati con i contenuti del registro H.

## SPHL

**SPHL** (Spostare HL a SP)
(SP) $\leftarrow$ (H) (L)
I contenuti dei registri H ed L (16 bit) sono spostati nel registro SP.

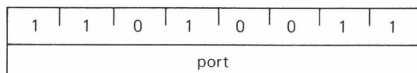
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

L'istruzione SPHL viene usata per caricare il registro stack pointer con i contenuti della coppia di registri H, L. I contenuti di L vengono posti negli otto bit LO dello stack pointer, e i contenuti di H sono posti negli otto bit H dello stack pointer. Come rilevato nel Manuale NEC Microcomputers, Inc: "L'istruzione SPHL può essere usata per caricare lo stack pointer con un valore che è stato elaborato usando le operazioni aritmetiche su due registri disponibili con la coppia di registri HL. Questo andrebbe sempre fatto con attenzione, dato che è facile perdere la traccia di dove lo stack pointer sta puntando, con conseguente perdita del contenuto dello stack."

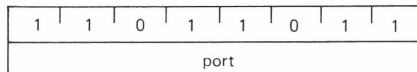
## OUT port

L'istruzione OUT port sposta i contenuti a 8 bit dell'accumulatore nella porta di uscita specificata dal secondo byte dell'istruzione.

Possono essere selezionate 256 porte di uscita. Durante il terzo ciclo macchina dell'istruzione, il codice dispositivo appare sul bus d'indirizzo, viene generato un impulso di controllo $\overline{\text{OUT}}$, e i contenuti dell'accumulatore appaiono sul bus di dati bidirezionale esterno.

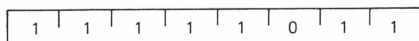
**OUT port** (Uscita)
(data) ← (A)
Il contenuto del registro A è posto sul bus di dati bidirezionale a 8 bit per la trasmissione alla porta specificata.

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

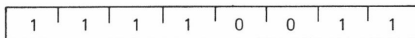
## IN port

**IN port** (Ingresso)
(A) ← (data)
I dati posti sul bus di dati bidirezionale a otto bit dalla porta specificata, sono spostati nel registro A.

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

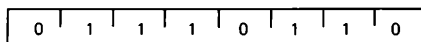
L'istruzione IN port permette al chip 8080A di leggere i dati presenti alla porta d'ingresso data dal secondo byte dell'istruzione. Possono essere indirizzate 256 porte d'ingresso. Durante il terzo ciclo macchina dell'istruzione, il codice dispositivo per il dispositivo di ingresso appare sul bus d'indirizzo, sul bus di controllo appare un segnale di controllo $\overline{\text{IN}}$, e e l'informazione che appare sul bus di dati bidirezionale appare anche nell'accumulatore.

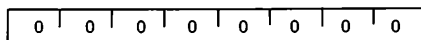
## EI e DI

**EI** (Abilitare le interruzioni)
Il sistema di interruzione è abilitato
**dopo l'esecuzione dell'istruzione successiva.**

Cicli: 1
Stati: 4
Flag: nessuno

**DI** (Disabilitare le interruzioni)
Il sistema di interruzione è disabilitato
**immediatamente dopo l'esecuzione dell'istruzione DI.**

Cicli:1
Stati: 4
Flag: nessuno

Citiamo il *μCOM-8 Software Manual* della NEC Microcomputers, Inc.: "Il fatto che l'8080A risponda o meno ad una richiesta d'interruzione è determinato dallo stato di un flip-flop interno di interrupt, INTE. Quando questo flip-flop è posizionato su uno, il processore risponde alle interruzioni. Quando viene resettato a zero, il processore ignora le richieste d'interruzione. Il flip-flop INTE è coinvolto sia dal controllo di programma che dall'operazione di sistema. Le operazioni di sistema che coinvolgono INTE sono un reset del sistema e il riconoscimento di un'interruzione. Entrambe le operazioni azzerano INTE e disabilitano così l'abilitazione all'interruzione.
Se devono essere rilevate altre interruzioni dopo un reset o un riconoscimento d'interruzione, il programma deve riabilitare il flip-flop. Due istruzioni, EI (Enable Interrupt) e DI (Disable Interrupt), forniscono il controllo programmato del flip-flop INTE. L'istruzione EI posiziona il flip-flop INTE su uno, abilitando l'interruzione, mentre l'istruzione DI azzera il flip-flop INTE, disabilitando l'interruzione. Lo stesso accade se si desidera che una sezione del programma venga eseguita alla massima velocità e senza possibilità d'interruzione, l'istruzione DI può essere usata per disabilitare le interruzioni per quella sezione di codice. Dopo che la sezione è completa, EI riabilita l'interruzione. Dato che il riconoscimento di una richiesta d'interruzione resetta il flip-flop INTE su zero, in qualunque routine, la prima istruzione a servire le interruzioni dovrebbe essere una EI. (Questo presuppone che il riconoscimento dell'interruzione resetti la richiesta d'interruzione stessa. Ciò deve essere fatto per evitare che il processore 8080A abbia un arresto imprevisto). Bisognerebbe fare un'eccezione quando viene servito il dispositivo di I/O più veloce. Per evitare che questa unità di I/O venga disturbata, il flip-flop INTE dovrebbe essere abilitato alla fine della routine."

## HLT

L'istruzione HLT fa sì che il processore sospenda l'operazione finchè il chip 8080A riceve un segnale RESET o riceve un segnale di richiesta d'interruzione(INT). Il processore accetta la richiesta INT a preescindere dalla condizione del flip-flop interno di interrupt. Dopo che l'interruzione è stata elaborata, l'esecuzione dell'istruzione continua alla posizione successiva dopo il comando di Halt.

## NOP

L'istruzione NOP non fa assolutamente niente all'infuori di occupare una locazione in memoria e prelevare quattro stati durante l'esecuzione del programma. Viene usata per la messa a punto di programmi, nei quali le istruzioni NOP in più vengono poste in un programma per una modifica successiva. Quando in un programma vengono fatte delle cancellazioni, le istruzioni NOP dovrebbero essere inserite al loro posto.

* * *

Con l'aiuto del materiale dell'*"Intel 8080 Microcomputers Systems User's Manual"* e il *"µCOM-8 Software Manual"* della NEC Microcomputers, abbiamo fornito una descrizione minuziosa delle singole istruzioni del chip 8080A. Siamo grati sia alla Intel Corporation che alla NEC Microcomputers, Inc. per il gentile permesso concessoci di usare le informazioni dei loro manuali. Se lavorate seriamente sul chip 8080A, dovreste possedere entrambi i manuali.

## INSTRUCTION SET

Summary of Processor Instructions

| Mnemonic | Description | Instruction Code [1] $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | Clock [2] Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| $MOV_{r1,r2}$ | Move register to register | 0 | 1 | D | D | D | S | S | S | 5 |
| MOV M, r | Move register to memory | 0 | 1 | 1 | 1 | 0 | S | S | S | 7 |
| MOV r, M | Move memory to register | 0 | 1 | D | D | D | 1 | 1 | 0 | 7 |
| HLT | Halt | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| MVI r | Move immediate register | 0 | 0 | D | D | D | 1 | 1 | 0 | 7 |
| MVI M | Move immediate memory | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 10 |
| INR r | Increment register | 0 | 0 | D | D | D | 1 | 0 | 0 | 5 |
| DCR r | Decrement register | 0 | 0 | D | D | D | 1 | 0 | 1 | 5 |
| INR M | Increment memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 10 |
| DCR M | Decrement memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 10 |
| ADD r | Add register to A | 1 | 0 | 0 | 0 | 0 | S | S | S | 4 |
| ADC r | Add register to A with carry | 1 | 0 | 0 | 0 | 1 | S | S | S | 4 |
| SUB r | Subtract register from A | 1 | 0 | 0 | 1 | 0 | S | S | S | 4 |
| SBB r | Subtract register from A with borrow | 1 | 0 | 0 | 1 | 1 | S | S | S | 4 |
| ANA r | And register with A | 1 | 0 | 1 | 0 | 0 | S | S | S | 4 |
| XRA r | Exclusive Or register with A | 1 | 0 | 1 | 0 | 1 | S | S | S | 4 |
| ORA r | Or register with A | 1 | 0 | 1 | 1 | 0 | S | S | S | 4 |
| CMP r | Compare register with A | 1 | 0 | 1 | 1 | 1 | S | S | S | 4 |
| ADD M | Add memory to A | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| ADC M | Add memory to A with carry | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 7 |
| SUB M | Subtract memory from A | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 7 |
| SBB M | Subtract memory from A with borrow | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 7 |
| ANA M | And memory with A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 7 |
| XRA M | Exclusive Or memory with A | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 7 |
| ORA M | Or memory with A | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| CMP M | Compare memory with A | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 7 |
| ADI | Add immediate to A | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| ACI | Add immediate to A with carry | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 7 |
| SUI | Subtract immediate from A | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 7 |
| SBI | Subtract immediate from A with borrow | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 7 |
| ANI | And immediate with A | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 7 |
| XRI | Exclusive Or immediate with A | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 7 |
| ORI | Or immediate with A | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| CPI | Compare immediate with A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 7 |
| RLC | Rotate A left | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 4 |
| RRC | Rotate A right | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 4 |
| RAL | Rotate A left through carry | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 4 |
| RAR | Rotate A right through carry | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 4 |
| JMP | Jump unconditional | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 10 |
| JC | Jump on carry | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 10 |
| JNC | Jump on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 10 |
| JZ | Jump on zero | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 10 |
| JNZ | Jump on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 10 |
| JP | Jump on positive | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 10 |
| JM | Jump on minus | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 10 |
| JPE | Jump on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 10 |
| JPO | Jump on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 10 |
| CALL | Call unconditional | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 17 |
| CC | Call on carry | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 11/17 |
| CNC | Call on no carry | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 11/17 |
| CZ | Call on zero | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 11/17 |
| CNZ | Call on no zero | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 11/17 |
| CP | Call on positive | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 11/17 |
| CM | Call on minus | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 11/17 |
| CPE | Call on parity even | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 11/17 |
| CPO | Call on parity odd | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 11/17 |
| RET | Return | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 10 |
| RC | Return on carry | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 5/11 |
| RNC | Return on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 5/11 |
| RZ | Return on zero | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 5/11 |
| RNZ | Return on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5/11 |
| RP | Return on positive | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 5/11 |
| RM | Return on minus | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 5/11 |
| RPE | Return on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 5/11 |
| RPO | Return on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 5/11 |
| RST | Restart | 1 | 1 | A | A | A | 1 | 1 | 1 | 11 |
| IN | Input | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 10 |
| OUT | Output | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 10 |
| LXI B | Load immediate register Pair B & C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 10 |
| LXI D | Load immediate register Pair D & E | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 10 |
| LXI H | Load immediate register Pair H & L | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 10 |
| LXI SP | Load immediate stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 10 |
| PUSH B | Push register Pair B & C on stack | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 11 |
| PUSH D | Push register Pair D & E on stack | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 11 |
| PUSH H | Push register Pair H & L on stack | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 11 |
| PUSH PSW | Push A and Flags on stack | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 11 |
| POP B | Pop register pair B & C off stack | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 10 |
| POP D | Pop register pair D & E off stack | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 10 |
| POP H | Pop register pair H & L off stack | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 10 |
| POP PSW | Pop A and Flags off stack | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 10 |
| STA | Store A direct | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 13 |
| LDA | Load A direct | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 13 |
| XCHG | Exchange D & E, H & L Registers | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 4 |
| XTHL | Exchange top of stack, H & L | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 18 |
| SPHL | H & L to stack pointer | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 5 |
| PCHL | H & L to program counter | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 5 |
| DAD B | Add B & C to H & L | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 10 |
| DAD D | Add D & E to H & L | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 10 |
| DAD H | Add H & L to H & L | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 10 |
| DAD SP | Add stack pointer to H & L | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 10 |
| STAX B | Store A indirect | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 7 |
| STAX D | Store A indirect | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 7 |
| LDAX B | Load A indirect | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 7 |
| LDAX D | Load A indirect | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 7 |
| INX B | Increment B & C registers | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 5 |
| INX D | Increment D & E registers | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 5 |
| INX H | Increment H & L registers | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 5 |
| INX SP | Increment stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 5 |
| DCX B | Decrement B & C | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 5 |
| DCX D | Decrement D & E | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| DCX H | Decrement H & L | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 5 |
| DCX SP | Decrement stack pointer | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 5 |
| CMA | Complement A | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 4 |
| STC | Set carry | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 4 |
| CMC | Complement carry | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 4 |
| DAA | Decimal adjust A | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 4 |
| SHLD | Store H & L direct | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 16 |
| LHLD | Load H & L direct | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 16 |
| EI | Enable Interrupts | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 4 |
| DI | Disable interrupt | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| NOP | No-operation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |

NOTE: 1. DDD o SSS - 000 B - 001 C - 010 D - 011 E - 100 H - 101 L - 110 Memory - 111 A

2. Due possibili tempi di ciclo, (5/11) indicano che il ciclo dell'istruzione dipende dai flag di condizione

## INTRODUZIONE AGLI ESPERIMENTI

I seguenti esperimenti forniscono un numero di programmi interessanti di cui potete aver bisogno se lavorate con strumenti digitali.

| Esperimento N. | Commento |
|---|---|
| 1 | Dimostra l'esecuzione di una routine che converte un numero BCD a due cifre in un numero binario a 8 bit. |
| 2 | Dimostra l'esecuzione di un programma che esegue un'addizione BCD a 16 cifre di due numeri. Il risultato deve essere meno o uguale a 9.999.999.999.999.999. |
| 3 | Dimostra l'esecuzione di una routine che converte un numero binario a 16 bit in un numero di cinque cifre BCD |

## ESPERIMENTO N. 1

### Scopo

Lo scopo di questo esperimento é caricare ed eseguire la routine: "BCD input and direct conversion to binary" N. 80-147 nella Libreria Utenti della Intel Microcomputer. Il programma è stato sviluppato da M.H. Gansler.

### Programma

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 076 | MVI A | Poni in modo immediato il byte nell'accumulatore |
| 001 | * | — | Byte di dati a due cifre BCD che deve essere convertito in un numero binario a 8 bit. |
| 002 | 117 | MOV C,A | Poni i contenuti dell'accumulatore nel registro C |
| 003 | 346 | ANI | AND immediato del byte seguente, con il contenuto dell'accumulatore |
| 004 | 017 | 017 | Byte di mascheramento che rivela la cifra BCD più significativa |
| 005 | 137 | MOV E,A | Poni i contenuti dell'accumulatore nel registro E |
| 006 | 171 | MOV A,C | Poni i contenuti del registro C nell'accumulatore |
| 007 | 346 | ANI | AND immediato del byte seguente con il contenuto dell'accumulatore |
| 010 | 360 | 360 | Byte di mascheramento che rivela la cifra BCD meno significativa |
| 011 | 017 | RRC | Rotazione dei contenuti dell'accumulatore di un bit verso destra e nel flag di riporto |
| 012 | 017 | RRC | Come sopra |
| 013 | 117 | MOV C,A | Poni i contenuti dell'accumulatore nel registro C |
| 014 | 017 | RRC | Rotazione dei contenuti dell'accumulatore di un bit verso destra e nel flag di riporto |
| 015 | 017 | RRC | Come sopra |
| 016 | 201 | ADD C | Somma i contenuti del registro C ai contenuti dell'accumulatore |

| | | | |
|---|---|---|---|
| 017 | 007 | RLC | Rotazione dei contenuti dell'accumulatore di un bit verso sinistra e nel flag di riporto |
| 020 | 203 | ADD E | Somma i contenuti del registro E ai contenuti dell'accumulatore |
| 021 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore sulla porta di uscita data nel byte dell'istruzione seguente |
| 022 | 000 | 000 | Codice dispositivo per la porta di uscita O |
| 023 | 166 | HLT | Alt |

**Commento**

Il programma inizia con il numero BCD a due cifre nell'accumulatore. Il programma può essere una subroutine; può sostituire all'istruzione HLT un'istruzione RET all'indirizzo di memoria LO = 023. Il programma suddetto può essere posizionato in qualunque punto della memoria. Noi abbiamo posizionato l'origine del programma a HI = 003 e LO = 000.

**Passo 1**

Caricate ed eseguite il programma suddetto per il numero decimale a due cifre 56. Il BCD equivalente è 01010110, o 56 in codice esadecimale e 126 in codice ottale. Quale numero binario osservate?

Abbiamo osservato 00111000 in codice binario, o 070 in ottale.

**Passo 2**

Cambiate il numero BCD all'indirizzo di memoria HI = 003 e LO = 001 secondo i numeri dati nella tabella seguente. Confrontate i vostri risultati con i risultati che abbiamo osservato.

| Numero decimale | Numero binario osservato | Numero binario predetto |
|---|---|---|
| 1 | | 00000001 |
| 10 | | 00001010 |
| 20 | | 00010100 |
| 50 | | 00110010 |
| 75 | | 01001011 |
| 80 | | 01010000 |
| 90 | | 01011010 |
| 99 | | 01100011 |

Per confermare l'ultima conversione da BCD in binario,

99 = 1 + 2 + 32 + 64. Si, va bene.

## ESPERIMENTO N. 2

### Scopo

Lo scopo di questo esperimento è caricare ed eseguire una subroutine di addizione BCD a 16 cifre, nella quale due numeri BCD vengono addizionati insieme per produrre un risultato che è minore o uguale a 9.999.999.999.999.999. Questo programma è elencato e descritto in tutti i particolari nel *μCOM-8 Software Manual* e viene presentato qui per concessione della NEC Microcomputers, Inc. Il programma viene iniziato alla posizione di memoria HI = 003 e LO = 024.

### Programma

| | Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|---|
| | 024 | 021 | LXI D | Carica in modo immediato due byte nei registri E e D, rispettivamente |
| | 025 | 347 | – | I registri D ed E contengono l'indirizzo a 16 |
| | 026 | 003 | – | bit delle cifre meno significative nell'augendo |
| | 027 | 041 | LXI H | Carica in modo immediato due byte nei registri L ed H, rispettivamente |
| | 030 | 357 | – | I registri H ed L contengono l'indirizzo |
| | 031 | 003 | – | a 16 bit delle cifre meno significative nell'addendo |
| ADD16: | 032 | 365 | PUSH PSW | Inserisci la parola di stato del programma nello stack (NOTA: Accertatevi di aver caricato lo stack pointer prima di eseguire questo programma). |
| | 033 | 305 | PUSH B | Inserisci i contenuti della coppia di registri B,C nello stack |
| | 034 | 016 | MVI C | Poni in modo immediato il byte seguente nel registro C |
| | 035 | 010 | – | Numero binario uguale a una volta e mezza il numero delle cifre BCD. Così, per le cifre BCD 16, il codice ottale sarebbe 010. |
| | 036 | 257 | XRA A | Azzera l'accumulatore e il flag di riporto |
| LOOP2: | 037 | 032 | LDAX D | Carica l'accumulatore dalla locazione di memoria indirizzata dalla coppia di registri D,E. |
| | 040 | 216 | ADC M | Somma i contenuti della locazione di memoria indirizzata dalla coppia di registri H, L, ai contenuti dell'accumulatore. |

| | | | | |
|---|---|---|---|---|
| | 041 | 047 | DAA | Aggiustamento decimale dei contenuti dell'accumulatore |
| | 042 | 022 | STAX D | Memorizza i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D, E. |
| | 043 | 015 | DCR C | Decrementa di 1 i contenuti del registro C |
| | 044 | 312 | JZ | Salta alla posizione di memoria DONE2 se i contenuti del registro C sono zero |
| | 045 | 054 | – | Byte d'indirizzo LO di DONE2 |
| | 046 | 003 | – | Byte d'indirizzo HI di DONE2 |
| | 047 | 053 | DCX H | Decrementa di 1 i contenuti della coppia di registri H,L |
| | 050 | 033 | DCX D | Decrementa di 1 i contenuti della coppia di registri D, E |
| | 051 | 303 | JMP | Salta alla posizione di memoria LOOP2 |
| | 052 | 037 | – | Byte d'indirizzo LO di LOOP2 |
| | 053 | 003 | – | Byte d'indirizzo HI di LOOP2 |
| DONE2: | 054 | 301 | POP B | Estrai i contenuti della coppia di registri B,C dallo stack |
| | 055 | 361 | POP PSW | Estrai la parola di stato del programma dallo stack |
| | 056 | 172 | MOV A,D | Poni i contenuti del registro D nell'accumulatore |
| | 057 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore |
| | 060 | 001 | 001 | Codice dispositivo della porta 1 |
| | 061 | 173 | MOV A,E | Sposta i contenuti del registro E nell'accumulatore |
| | 062 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore |
| | 063 | 000 | 000 | Codice dispositivo della porta O |
| | 064 | 166 | HLT | Alt |

**Commento**

Questo programma inizia con un augendo BCD a 16 cifre negli indirizzi di memoria da 340 a 347, con la cifra BCD meno significativa nella posizione 347 e quella più significativa nella posizione 340.

L'addendo BCD a 16 cifre è inizialmente negli indirizzi di memoria LO da 350 a 357, con la cifra BCD meno significativa nella posizione 357 e quella più significativa nella posizione 350. Seguono le definizioni dei termini *"addendo"* e *"augendo"*.

*Augendo* — In un'addizione aritmetica, il numero che è aumentato poichè un altro numero (chiamato addendo) è stato addizionato ad esso.

*Addendo* — Una quantità che, se aggiunta ad un'altra quantità (chiamata augendo), produce un risultato detto somma.

L'esecuzione del programma inizia a HI = 003 e LO = 024. La somma prende il posto dell'augendo.

Considerate un augendo di 1.000.000.000.000.099 e un addendo di 8.000.000.000.000.001. La memoria per questi due numeri BCD a 16 cifre, si presenta nel modo seguente (tutto a HI = 003):

| Indirizzo di memoria LO | Cifre BCD | Codice ottale | Codice binario |
|---|---|---|---|
| 340 | 1.0 | 020 | 00010000 |
| 341 | 0.0 | 000 | 00000000 |
| 342 | 0.0 | 000 | 00000000 |
| 343 | 0.0 | 000 | 00000000 |
| 344 | 0.0 | 000 | 00000000 |
| 345 | 0.0 | 000 | 00000000 |
| 346 | 0.0 | 000 | 00000000 |
| 347 | 9.9 | 231 | 10011001 |
| 350 | 8.0 | 200 | 100000000 |
| 351 | 0.0 | 000 | 00000000 |
| 352 | 0.0 | 000 | 00000000 |
| 353 | 0.0 | 000 | 00000000 |
| 354 | 0.0 | 000 | 00000000 |
| 355 | 0.0 | 000 | 00000000 |
| 356 | 0.0 | 000 | 00000000 |
| 357 | 0.1 | 001 | 00000001 |

Quando questi due numeri sono addizionati, la somma 9.000.000.000.000.100 rimpiazza l'augendo nelle locazioni di memoria da HI = 003 e LO = 340 a HI = 003 e LO = 347.

**Passo 1**

Caricate in memoria il programma suddetto. Caricate l'augendo, 1.000.000.000.000.099, e l'addendo, 8.000.000.000.000.001, in memoria. Eseguite il programma. Qual'è la somma che appare iniziando all'indirizzo di memoria LO = 340?

Abbiamo osservato la seguente sequenza di numeri BCD in locazioni di memoria successive, iniziando a LO = 340:

90
00
00
00
00
00
01
00

che corrispondono al numero BCD a 16 cifre, 9.000.000.000.000.100.

**Passo 2**

Addizionate i seguenti numeri BCD e confrontate i vostri risultati con quelli che abbiamo osservato.

| Augendo | Addendo | Somma |
|---|---|---|
| 3.000.000.000.000.100 | 1.000.000.000.000.001 | 4.000.000.000.000.101 |
| 0.000.000.000.123.456 | 0.000.000.000.240.833 | 0.000.000.000.364.289 |
| 0.000.000.000.927.928 | 0.000.000.000.844.992 | 0.000.000.001.772.920 |
| 9.999.999.999.999.999 | 0.000.000.000.000.001 | 0.000.000.000.000.000 |

**Passo 3**

A LO = 035, cambiate il byte in 002 che corrisponde all'addizione di due numeri BCD a quattro cifre. Eseguite le seguenti addizioni:

0099 + 0001 = 0100

9999 + 0001 = 0000

0001 + 0001 = 0002

0023 + 0077 = 0100

## ESPERIMENTO N. 3

**Scopo**

Lo scopo di questo esperimento è caricare ed eseguire la routine di conversione binario - BCD N. 80-67 della Libreria Utenti della Intel Microcomputer. Il programma è stato sviluppato da Niels S, Gundestrup del Laboratorio Isotopico Geofisico in Danimarca.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 222 | 021 | LXI D | Poni in modo immediato due byte nella coppia di registri D. Questo è il numero binario a 16 bit che sarà convertito in un numero a 5 cifre BCD. |
| 223 | * | – | 8 bit meno significativi del numero binario a 16 bit |
| 224 | * | – | 8 bit più significativi del numero binario a 16 bit |
| 225 | 041 | LXI H | Poni in modo immediato due byte nella coppia di registri H. Questo è l'indirizzo di memoria della cifra più significativa (MSD) dei numeri BCD a 5 cifre. Le restanti quattro cifre sono memorizzate in locazioni di memoria successive, una cifra per ogni locazione. |
| 226 | 340 | – | Byte del registro L |
| 227 | 003 | – | Byte del registro H |
| BNCCD: 230 | 365 | PUSH PSW | Inserisci i contenuti della parola di stato del programma nello stack |
| 231 | 305 | PUSH B | Inserisci i contenuti della coppia di registri B nello stack |
| 232 | 325 | PUSH D | Inserisci i contenuti della coppia di registri D nello stack |
| 233 | 345 | PUSH H | Inserisci i contenuti della coppia di registri H nello stack |
| 234 | 353 | XCHG | Scambia i contenuti della coppia di registri H con i contenuti della coppia di registri D. |
| 235 | 001 | LXI B | Poni in modo immediato due byte nella coppia i registri B. (10.000) |
| 236 | 360 | – | Byte del registro C |
| 237 | 330 | – | Byte del registro B |
| 240 | 315 | CALL | Richiama la subroutine DECNO, che esegue la conversione da binario a BCD (MSD). |

| | | | |
|---|---|---|---|
| 241 | 276 | – | Byte d'indirizzo LO |
| 242 | 003 | – | Byte d'indirizzo HI |
| 243 | 001 | LXI B | Poni in modo immediato due byte nella coppia di registri B. (1000) |
| 244 | 030 | – | Byte del registro C |
| 245 | 374 | – | Byte del registro B |
| 246 | 315 | CALL | Richiama la subroutine DECNO |
| 247 | 276 | – | Byte d'indirizzo LO |
| 250 | 003 | – | Byte d'indirizzo HI |
| 251 | 001 | LXI B | Poni in modo immediato due byte nella coppia di registri B. (100) |
| 252 | 234 | – | Byte del registro C |
| 253 | 377 | – | Byte del registro B |
| 254 | 315 | CALL | Richiama la subroutine DECNO |
| 255 | 276 | – | Byte d'indirizzo LO |
| 256 | 003 | – | Byte d'indirizzo HI |
| 257 | 001 | LXI B | Poni in modo immediato due byte nella coppia di registri B. (10) |
| 260 | 366 | – | Byte del registro C |
| 261 | 377 | – | Byte del registro B |
| 262 | 315 | CALL | Richiama la subroutine DECNO |
| 263 | 276 | – | Byte d'indirizzo LO |
| 264 | 003 | – | Byte d'indirizzo HI |
| 265 | 175 | MOV A, L | Sposta i contenuti del registro L nell'accumulatore |
| 266 | 306 | ADI | Somma in modo immediato il byte ai contenuti dell'accumulatore |
| 267 | 000 | 000 | (NOTA: 260 è il codice ASCII se lo si desidera) |
| 270 | 022 | STAX D | Memorizza i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D |
| 271 | 341 | POP H | Estrai la coppia di registri H dallo stack |
| 272 | 321 | POP D | Estrai la coppia di registri D dallo stack |

| | | | |
|---|---|---|---|
| 273 | 301 | POP B | Estrai la coppia di registri B dallo stack |
| 274 | 361 | POP PSW | Estrai la parola di stato del progr. dallo stack |
| 275 | 311 | RET | Rientro dalla subroutine |
| DECNO: 276 | 076 | MVI A | Azzera i contenuti dell'accumulatore |
| 277 | 000 | 000 | (NOTA: 260 è il codice ASCII se lo si desidera) |
| 300 | 325 | PUSH D | Inserisci il registro D sullo stack |
| 301 | 135 | MOV E, L | Sposta i contenuti del registro L nel registro E |
| 302 | 124 | MOV D,H | Sposta i contenuti del registro H nel registro D |
| 303 | 074 | INR A | Incrementa i contenuti dell'accumulatore di 1 |
| 304 | 011 | DAD B | Somma i contenuti della coppia di registri B ai contenuti della coppia di registri H e memorizza nella coppia di registri H |
| 305 | 332 | JC | Salta se il flag di riporto è a livello logico 1 |
| 306 | 301 | – | Byte d'indirizzo LO |
| 307 | 003 | – | Byte d'indirizzo HI |
| 310 | 075 | DCR A | Decrementa i contenuti dell'accumulatore di 1 |
| 311 | 153 | MOV L,E | Sposta i contenuti del registro E nel registro L |
| 312 | 142 | MOV H,D | Sposta i contenuti del registro D nel registro H |
| 313 | 321 | POP D | Estrai la coppia di registri D dallo stack |
| 314 | 022 | STAX D | Memorizza i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D |
| 315 | 023 | INX D | Incrementa di 1 i contenuti della coppia di registri D |
| 316 | 311 | RET | Rientro alla subroutine DECNO |

**Commento**

Questo programma inizia con un numero binario a 16 bit nella coppia di registri D,E. Il numero è convertito in un numero a quattro cifre BCD che viene memorizzato iniziando da HI = 003 e LO = 340. La cifra BCD più significativa è memorizzata in questa locazione, e le restanti quattro cifre nelle locazioni successive. La cifra BCD meno significativa è memorizzata a LO = 344. Il programma BNBCD inizia da HI = 003 e LO = 230; comunque, il numero binario a 16 bit si deve trovare nella coppia di registri D, e la locazione della cifra più significativa nella coppia di registri H. Abbiamo usato istruzioni LXI per settare questa informazione nei registri prima che BNBCD venga eseguito.

L'uscita può essere sia sotto forma di numerali decimali o di codice ASCII a 8 bit, con il bit più significativo (il bit di parità) a livello logico 1. Nel programma originale è stato corretto un piccolo errore per permettere ad LSD di essere memorizzata nel codice ASCII.

Nelle pagine seguenti è mostrato il contributo originale alla Libreria Utenti della Intel. Notate lo stile del programma in "cross-assembler", i commenti al programma (che seguono su ogni riga il punto e virgola), e il fatto che sia gli indirizzi di memoria che i byte istruzione sono elencati in codice esadecimale. Questo elenco vi dà tracce importanti per il funzionamento del programma.

**Passo 1**

Caricate il programma in memoria. Posizionate i byte su 000 a LO = 267 e LO = 277. Caricate 377 sia su LO = 223 che su LO = 224. Questi due byte istruzioni corrispondono al numero binario a 16 bit, 1111111111111111, che ha un valore decimale di 65.535.

**Passo 2**

Eseguite il programma. Risettate il microcomputer e determinate i contenuti delle locazioni di memoria da LO = 340 a LO = 344. Quale sequenza di numeri decimali osservate in queste locazioni?

Abbiamo osservato 6 5 5 3 5, come previsto.

**Passo 3**

Determinate i BCD a cinque cifre equivalenti ai seguenti numeri binari a 16 bit, che dovrebbero essere caricati nelle locazioni di memoria LO = 233 e LO = 224 prima di eseguire il programma. Controllate i vostri risultati con i nostri.

| Byte del registro D (LO = 224) | Byte del registro E (LO = 223) | Numero BCD a 5 cifre osservato |
|---|---|---|
| 377 | 377 | 65535 |
| 200 | 000 | 32768 |
| 100 | 000 | 16384 |
| 040 | 000 | 08192 |
| 020 | 000 | 04096 |
| 010 | 000 | 02048 |
| 004 | 000 | 01024 |
| 002 | 000 | 00512 |
| 001 | 000 | 00256 |
| 000 | 200 | 00128 |
| 000 | 100 | 00064 |
| 000 | 040 | 00032 |
| 000 | 020 | 00016 |
| 000 | 010 | 00008 |
| 000 | 004 | 00004 |
| 000 | 002 | 00002 |
| 000 | 001 | 00001 |
| 000 | 000 | 00000 |

**Passo 4**

Un programma che converte una parola binaria a 16 bit in un numero BCD a 5 cifre è molto utile. Diamo qui sotto un elenco in codice esadecimale di un programma che inizia a HI = 001 e LO = 000. Lo abbiamo caricato nella EPROM e possiamo usarlo come una subroutine.

| Indirizzo di memoria LO | Byte di istruzione | Indirizzo di memoria LO | Byte di istruzione |
|---|---|---|---|
| BNBCD: 00 | F5 | 1A | CD |
| 01 | C5 | 1B | 24 |
| 02 | D5 | 1C | 01 |
| 03 | E5 | 1D | 7D |
| 04 | EB | 1E | 12 |
| 05 | 01 | 1F | E1 |
| 06 | F0 | 20 | D1 |
| 07 | D8 | 21 | C1 |
| 08 | CD | 22 | F1 |
| 09 | 24 | 23 | C9 RET |
| 0A | 01 | DECNO: 24 | AF |
| 0B | 01 | 25 | D5 |
| 0C | 18 | 26 | 5D |
| 0D | FC | 27 | 54 |
| 0E | CD | 28 | 3C |
| 0F | 24 | 29 | 09 |
| 10 | 01 | 2A | DA |
| 11 | 01 | 2B | 26 |
| 12 | 9C | 2C | 01 |
| 13 | FF | 2D | 3D |
| 14 | CD | 2E | 6B |
| 15 | 24 | 2F | 62 |
| 16 | 01 | 30 | D1 |
| 17 | 01 | 31 | 12 |
| 18 | F6 | 32 | 13 |
| 19 | FF | 33 | C9 RET |

```
                   ;BINARY TO BCD SUBROUTINE

                   ;INPUT:  UNSIGNED BINARY NUMBER IN D,E
                   ;        POINTER TO LOWEST BUFFER LOC IN HL

                   ;OUTPUT: 5 BCD-DIGITS, ONE DIGIT PER MEMORY LOC.
                   ;        HL POINT TO MSD IN LOWEST LOCATION.

0100 F5            BNBCD:   PUSH PSW         ;SAVE VARIABLES
0101 C5                     PUSH B
0102 D5                     PUSH D
0103 E5                     PUSH H
0104 EB                     XCHG             ;GET NUMBER IN HL, ADDR IN DE
0105 01F0D8                 LXI B,-10000
0108 CD2401                 CALL DECNO       ;GET MSD
010B 0118FC                 LXI B,-1000
010E CD2401                 CALL DECNO
0111 019CFF                 LXI B,-100
0114 CD2401                 CALL DECNO
0117 01F6FF                 LXI B,-10
011A CD2401                 CALL DECNO
011D 7D                     MOV A,L          ;GET LSD
011E 12                     STAX D           ;STORE IT
011F E1                     POP H
0120 D1                     POP D
0121 C1                     POP B
0122 F1                     POP PSW
0123 C9                     RET

0124 AF            DECNO:   XRA A            ;0 TO A. USE 30H IF ASCII
0125 D5                     PUSH D           ;SAVE ADDR
0126 5D                     MOV E,L          ;SAVE BINARY
0127 54                     MOV D,H
0128 3C                     INR A            ;INCRENENT DIGIT
0129 09                     DAD B            ;SUBTRACT
012A DA2601                 JC DECNO+2       ;RESULT NEGATIVE?
012D 3D                     DCR A            ;YES, RESTORE DIGIT COUNT
012E 6B                     MOV L,E          ;BINARY NUMBER
012F 62                     MOV H,D
0130 D1                     POP D            ;AND ADDRESS
0131 12                     STAX D           ;STORE DIGIT
0132 13                     INX D            ;INCREMENT POINTER
0133 C9                     RET
```

Test program for binary to BCD conversion

Using the monitor:

1. Deposit binary value in DE using X-command

2. Type G50
BCD in 1001-1005.

2

```
0000             PROM     EQU 0
FFFF             RAM      EQU NOT PROM
10FF             STPNT    EQU 10FFH

0000                      ORG 50H
0050 31FF10               LXI SP,STPNT
0053 210110               LXI H,1001H
0056 CD0001               CALL BNBCD
0059 CF                   RST 1
005A C35600               JMP $-4


005D                      ORG 100H
```

## ELENCO OTTALE/ESADECIMALE DEL SET DI ISTRUZIONI 8080

Nelle pagine seguenti, formiamo un ampio elenco dei 256 codici di istruzione del set di istruzioni del microprocessore 8080.
Questo elenco fornisce le seguenti informazioni:

- Il codice istruzioni, in ottale
- Il codice istruzioni, in esadecimale
- Il codice istruzioni, nel codice mnemonico della Intel Corporation
- Una breve descrizione di ciò che fa il codice istruzioni.

Volendo, potete fare una fotocopia di questo elenco e tenerlo a portata di mano, Noi siamo del parere che esso abbia un particolare valore quando tentiamo di convertire un elenco ottale in esadecimale, o viceversa.

Seguendo l'elenco ottale/esadecimale, forniamo anche un sommario di una pagina del set di istruzioni 8080, realizzato a seconda del numero di byte nell'istruzione.

| Ottale | <B1> Esadecimale | Mnemonico | Descrizione |
|---|---|---|---|
| 000 | 00 | NOP | Nessuna operazione |
| 001 | 01 | LXI B <B2> <B3> | Carica in modo immediato nella coppia di registri B e C |
| 002 | 02 | STAX B | Memorizza A in modo indiretto in M indirizzato da B e C |
| 003 | 03 | INX B | Incrementa di 1 i contenuti della coppia di registri B e C |
| 004 | 04 | INR B | Incrementa di 1 il registro B |
| 005 | 05 | DCR B | Decrementa di 1 il registro B |
| 006 | 06 | MVI B <B2> | Sposta in modo immediato nel registro B |
| 007 | 07 | RLC | Ruota A verso sinistra |
| 010 | 08 | – | – – |
| 011 | 09 | DAD B | Somma i contenuti di B,C ad H,L e memorizza in H,L |
| 012 | 0A | LDAX B | Carica A in modo indiretto da M indirizzato da B e C |
| 013 | 0B | DCX B | Decrementa di 1 i contenuti della coppia di registri B e C |
| 014 | 0C | INR C | Incrementa di 1 il registro C |
| 015 | 0D | DCR C | Decrementa di 1 il registro C |
| 016 | 0E | MVI C <B2> | Sposta in modo immediato nel registro C |
| 017 | 0F | RRC | Ruota A verso destra |
| 020 | 10 | – | –– |
| 021 | 11 | LXI D <B2> <B3> | Carica in modo immediato nella coppia di registri D e E |

| | <B1> | | |
|---|---|---|---|
| 022 | 12 | STAX D | Memorizza A in modo indiretto in M indirizzato da D ed E |
| 023 | 13 | INX D | Incrementa di 1 i contenuti della coppia di registri D ed E |
| 024 | 14 | INR D | Incrementa di 1 il registro D |
| 025 | 15 | DCR D | Decrementa di 1 il registro D |
| 026 | 16 | MVI D <B2> | Sposta in modo immediato nel registro D |
| 027 | 17 | RAL | Ruota A a sinistra attraverso il carry |
| 030 | 18 | – | —— |
| 031 | 19 | DAD D | Somma i contenuti di D,E ad H,L e memorizza in H,L |
| 032 | 1A | LDAX D | Carica A in modo indiretto da M indirizzato da D ed E |
| 033 | 1B | DCX D | Decrementa di 1 i contenuti della coppia di registri D ed E |
| 034 | 1C | INR E | Incrementa di 1 il registro E |
| 035 | 1D | DCR E | Decrementa di 1 il registro E |
| 036 | 1E | MVI E <B2> | Sposta in modo immediato nel registro E |
| 037 | 1F | RAR | Ruota A verso destra attraverso il carry |
| 040 | 20 | – | —— |
| 041 | 21 | LXI H <B2> <B3> | Carica in modo immediato nella coppia di registri H ed L |
| 042 | 22 | SHLD <B2> <B3> | Memorizza L e H in M e M+1, dove M = <B2> <B3> |
| 043 | 23 | INX H | Incrementa di 1 i contenuti della coppia di registri H ed L |
| 044 | 24 | INR H | Incrementa di 1 il registro H |
| 045 | 25 | DCR | Decrementa di 1 il registro H |
| 046 | 26 | MVI H <B2> | Sposta in modo immediato nel registro H |
| 047 | 27 | DAA | Aggiustamento decimale di A |
| 050 | 28 | – | —— |
| 051 | 29 | DAD H | Somma i contenuti di H,L ad H,L e memorizzare in H,L |
| 052 | 2A | LHLD <B2> <B3> | Carica L ed H con i contenuti di M e M+1, rispettivamente |
| 053 | 2B | DCX H | Decrementa di 1 i contenuti della coppia di registri H ed L |
| 054 | 2C | INR L | Incrementa di 1 il registro L |
| 055 | 2D | DCR L | Decrementa di 1 il registro L |
| 056 | 2E | MVI L <B2> | Sposta in modo immediato nel registro L |
| 057 | 2F | CMA | Complementa A |
| 060 | 30 | – | —— |
| 061 | 31 | LXI SP <B2> <B3> | Carica in modo immediato nello stack pointer |
| 062 | 32 | STA <B2> <B3> | Memorizza A in modo diretto in M indirizzato da <B2> <B3> |
| 063 | 33 | INX SP | Incrementa di 1 il registro SP |
| 064 | 34 | INR M | Incrementa di 1 i contenuti di M |
| 065 | 35 | DCR M | Decrementa di 1 i contenuti di M |
| 066 | 36 | MVI M <B2> | Sposta in modo immediato in M indirizzato da H ed L |
| 067 | 37 | STC | Setta il flip-flop di carry al livello logico 1 |

| <B1> | | | |
|---|---|---|---|
| 070 | 38 | – | –– |
| 071 | 39 | DAD SP | Somma i contenuti dello stack pointer ad H,L e memorizza in H,L |
| 072 | 3A | LDA <B2> <B3> | Carica A in modo diretto con i contenuti di M indirizzati da <B2> <B3> |
| 073 | 3B | DCX SP | Decrementa di 1 il registro SP |
| 074 | 3C | INR A | Incrementa di 1 il registro A |
| 075 | 3D | DCR A | Decrementa di 1 il registro A |
| 076 | 3E | MVI A <B2> | Sposta in modo immediato nel registro A |
| 077 | 3F | CMC | Complementa il flip-flop di carry |
| 100 | 40 | MOV B,B | Sposta i contenuti del registro B nel registro B |
| 101 | 41 | MOV B,C | Sposta i contenuti del reg. C nel reg. B |
| 102 | 42 | MOV B,D | Sposta i contenuti del reg. D nel reg. B |
| 103 | 43 | MOV B,E | Sposta i contenuti del reg. E nel reg. B |
| 104 | 44 | MOV B,H | Sposta i contenuti del reg. H nel reg. B |
| 105 | 45 | MOV B,L | Sposta i contenuti del reg. L nel reg. B |
| 106 | 46 | MOV B,M | Sposta i contenuti di M nel reg. B |
| 107 | 47 | MOV B,A | Sposta i contenuti del reg. A nel reg. B |
| 110 | 48 | MOV C,B | Sposta i contenuti del reg. B nel reg. C |
| 111 | 49 | MOV C,C | Sposta i contenuti del reg. C nel reg. C |
| 112 | 4A | MOV C,D | Sposta i contenuti del reg. D nel reg. C |
| 113 | 4B | MOV C,E | Sposta i contenuti del reg E nel reg. C |
| 114 | 4C | MOV C,H | Sposta i contenuti del reg. H nel reg. C |
| 115 | 4D | MOV C,L | Sposta i contenuti del reg. L nel reg. C |
| 116 | 4E | MOV C,M | Sposta i contenuti di M nel reg. C |
| 117 | 4F | MOV C,A | Sposta i contenuti del reg. A nel reg. C |
| 120 | 50 | MOV D,B | Sposta i contenuti del reg. B nel reg. D |
| 121 | 51 | MOV D,C | Sposta i contenuti del reg. C nel reg. D |
| 122 | 52 | MOV D,D | Sposta i contenuti del reg. D nel reg. D |
| 123 | 53 | MOV D,E | Sposta i contenuti del reg. E nel reg. D |
| 124 | 54 | MOV D,H | Sposta i contenuti del reg. H nel reg. D |
| 125 | 55 | MOV D,L | Sposta i contenuti del reg. L nel reg. D |
| 126 | 56 | MOV D,M | Sposta i contenuti di M nel reg. D |
| 127 | 57 | MOV D,A | Sposta i contenuti del reg. A nel reg. D |
| 130 | 58 | MOV E,B | Sposta i contenuti del reg. B nel reg. E |
| 131 | 59 | MOV E,C | Sposta i contenuti del reg. C nel reg. E |
| 132 | 5A | MOV E,D | Sposta i contenuti del reg. D nel reg. E |
| 133 | 5B | MOV E,E | Sposta i contenuti del reg. E nel reg. E |
| 134 | 5C | MOV E,H | Sposta i contenuti del reg. H nel reg. E |
| 135 | 5D | MOV E,L | Sposta i contenuti del reg. L nel reg. E |
| 136 | 5E | MOV E,M | Sposta i contenuti di M nel reg. E |
| 137 | 5F | MOV E,A | Sposta i contenuti del reg. A nel reg. E |
| 140 | 60 | MOV H,B | Sposta i contenuti del reg. B nel reg. H |
| 141 | 61 | MOV H,C | Sposta i contenuti del reg. C nel reg. H |
| 142 | 62 | MOV H,D | Sposta i contenuti del reg. D nel reg. H |
| 143 | 63 | MOV H,E | Sposta i contenuti del reg. E nel reg. H |
| 144 | 64 | MOV H,H | Sposta i contenuti del reg. H nel reg. H |
| 145 | 65 | MOV H,L | Sposta i contenuti del reg. L nel reg. H |
| 146 | 66 | MOV H,M | Sposta i contenuti di M nel reg. H |
| 147 | 67 | MOV H,A | Sposta i contenuti del reg. A nel reg. H |
| 150 | 68 | MOV L,B | Sposta i contenuti del reg. B nel reg. L |

<B1>

| | | | |
|---|---|---|---|
| 151 | 69 | MOV L,C | Sposta i contenuti del reg. C nel reg. L |
| 152 | 6A | MOV L,D | Sposta i contenuti del reg. D nel reg. L |
| 153 | 6B | MOV L,E | Sposta i contenuti del reg. B nel reg. L |
| 154 | 6C | MOV L,H | Sposta i contenuti del reg. H nel reg. L |
| 155 | 6D | MOV L,L | Sposta i contenuti del reg. L nel reg. L |
| 156 | 6E | MOV L,M | Sposta i contenuti di M nel reg. L |
| 157 | 6F | MOV L,A | Sposta i contenuti del reg. A nel reg. L |
| 160 | 70 | MOV M,B | Sposta i contenuti del reg. B in M |
| 161 | 71 | MOV M,C | Sposta i contenuti del reg. C in M |
| 162 | 72 | MOV M,D | Sposta i contenuti del reg. D in M |
| 163 | 73 | MOV M,E | Sposta i contenuti del reg. E in M |
| 164 | 74 | MOV M,H | Sposta i contenuti del reg. H in M |
| 165 | 75 | MOV M,L | Sposta i contenuti del reg. L in M |
| 166 | 76 | HLT | Alt |
| 167 | 77 | MOV M,A | Sposta i contenuti del reg. A in M |
| 170 | 78 | MOV A,B | Sposta i contenuti del reg. B nel reg. A |
| 171 | 79 | MOV A,C | Sposta i contenuti del reg. C nel reg. A |
| 172 | 7A | MOV A,D | Sposta i contenuti del reg. D nel reg. A |
| 173 | 7B | MOV A,E | Sposta i contenuti del reg. E nel reg. A |
| 174 | 7C | MOV A,H | Sposta i contenuti del reg. H nel reg. A |
| 175 | 7D | MOV A,L | Sposta i contenuti del reg. L nel reg. A |
| 176 | 7E | MOV A,M | Sposta i contenuti di M nel reg. A |
| 177 | 7F | MOV A,A | Sposta i contenuti del reg. A nel reg. A |
| 200 | 80 | ADD B | Somma i contenuti del reg. B al reg. A |
| 201 | 81 | ADD C | Somma i contenuti del reg. C al reg. A |
| 202 | 82 | ADD D | Somma i contenuti del reg. D al reg. A |
| 203 | 83 | ADD E | Somma i contenuti del reg. E al reg. A |
| 204 | 84 | ADD H | Somma i contenuti del reg. H al reg. A |
| 205 | 85 | ADD L | Somma i contenuti del reg. L al reg. A |
| 206 | 86 | ADD M | Somma i contenuti di M al reg. A |
| 207 | 87 | ADD A | Somma i contenuti del reg. A al reg. A |
| 210 | 88 | ADC B | Somma il carry e i cont. del reg. B al reg. A |
| 211 | 89 | ADC C | Somma il carry e i cont. del reg. C al reg. A |
| 212 | 8A | ADC D | Somma il carry e i cont. del reg. D al reg. A |
| 213 | 8B | ADC E | Somma il carry e i cont. del reg. E al reg. A |
| 214 | 8C | ADC H | Somma il carry e i cont. del reg. H al reg. A |
| 215 | 8D | ADC L | Somma il carry e i cont. del reg. L al reg. A |
| 216 | 8E | ADC M | Somma il carry e i cont di M al reg. A |
| 217 | 8F | ADC A | Somma il carry e i cont. del reg. A al reg. A |
| 220 | 90 | SUB B | Sottrai i cont. del reg. B dal reg. A |
| 221 | 91 | SUB C | Sottrai i cont. del reg. C dal reg. A |
| 222 | 92 | SUB D | Sottrai i cont. del reg. D dal reg. A |
| 223 | 93 | SUB E | Sottrai i cont. del reg. B dal reg. A |
| 224 | 94 | SUB H | Sottrai i cont. del reg. H dal reg. A |
| 225 | 95 | SUB L | Sottrai i cont. del reg. L dal reg. A |
| 226 | 96 | SUB M | Sottrai i cont. di M dal reg. A |
| 227 | 97 | SUB A | Azzera il registro A |
| 230 | 98 | SBB B | Sottrai il carry e i cont. del reg. B da A |
| 231 | 99 | SBB C | Sottrai il carry e i cont. del reg. C da A |
| 232 | 9A | SBB D | Sottrai il carry e i cont. del reg. D da A |
| 233 | 9B | SBB E | Sottrai il carry e i cont. del reg. E da A |
| 234 | 9C | SBB H | Sottrai il carru e i cont. del reg. H da A |
| 235 | 9D | SBB L | Sottrai il carry e i cont. del reg. L da A |

<B1>

| | | | |
|---|---|---|---|
| 236 | 9E | SBB M | Sottrai il carry e i cont. di M dal reg. A |
| 237 | 9F | SBB A | Sottrai il carry e i cont. del reg. A da A |
| 240 | A0 | ANA B | AND cont. del reg. B con il reg. A |
| 241 | A1 | ANA C | AND cont. del reg. C con il reg. A |
| 242 | A2 | ANA D | AND cont. del reg. D con il reg. A |
| 243 | A3 | ANA E | AND cont. del reg. E con il reg. A |
| 244 | A4 | ANA H | AND cont. del reg. H con il reg. A |
| 245 | A5 | ANA L | AND cont. del reg. L con il reg. A |
| 246 | A6 | ANA M | AND cont. di M con il reg. A |
| 247 | A7 | ANA A | AND cont. del reg. A con il reg. A |
| 250 | A8 | XRA B | OR escusivo cont. del reg. B con il reg. A |
| 251 | A9 | XRA C | OR esclusivo cont. del reg. C con il reg. A |
| 252 | AA | XRA D | OR esclusivo cont. del reg. D con il reg. A |
| 253 | AB | XRA E | OR esclusivo cont. del reg. E con il reg. A |
| 254 | AC | XRA H | OR esclusivo cont. del reg. H con il reg. A |
| 255 | AD | XRA L | OR esclusivo del reg. L con il reg. A |
| 256 | AE | XRA M | OR esclusivo cont. di M con il reg. A |
| 257 | AF | XRA A | OR esclusivo cont. del reg. A con il reg. A |
| 260 | B0 | ORA B | OR cont. del reg. B con il reg. A |
| 261 | B1 | ORA C | OR cont del reg. C con il reg. A |
| 262 | B2 | ORA D | OR cont. del reg. D con il reg. A |
| 263 | B3 | ORA E | OR cont. del reg. E con il reg. A |
| 264 | B4 | ORA H | OR cont. del reg. H con il reg. A |
| 265 | B5 | ORA L | OR cont. del reg. L con il reg. A |
| 266 | B6 | ORA M | OR cont. di M con il reg. A |
| 267 | B7 | ORA A | OR cont. del reg. A con il reg. A |
| 270 | B8 | CMP B | Confronta i cont. del reg. B con il reg. A |
| 271 | B9 | CMP C | Confronta i cont. del reg. C con il reg. A |
| 272 | BA | CMP D | Confronta i cont. del reg. D con il reg. A |
| 273 | BB | CMP E | Confronta i cont. del reg. E con il reg. A |
| 274 | BC | CMP H | Confronta i cont. del reg. H con il reg. A |
| 275 | BD | CMP L | Confronta i cont. del reg. L con il reg. A |
| 276 | BE | CMP M | Confronta i cont. di M con il reg. A |
| 277 | BF | CMP A | Confronta i cont. del reg. A con il reg. A |
| 300 | C0 | RNZ | Rientra dalla subroutine se il flip-flop di zero = livello logico 0 |
| 301 | C1 | POP B | Estrai dallo stack e memorizza l'indirizzo nella coppia di registri B e C |
| 302 | C2 | JNZ <B2> <B3> | Salta, se il flip-flop di zero = livello logico 0 |
| 303 | C3 | JMP <B2> <B3> | Salto incondizionato ad M indirizzato da <B2> <B3> |
| 304 | C4 | CNZ <B2> <B3> | Richiama la subroutine se il flip-flop di zero = livello logico 0 |
| 305 | C5 | PUSH B | Inserisci i contenuti della coppia di registri B e C nello stack |
| 306 | C6 | ADI <B2> | Somma in modo immediato al registro A |
| 307 | C7 | RST 0 | Richiama la suboutine all'indirizzo $000_8$ |
| 310 | C8 | RZ | Rientra dalla subroutine se il flip-flop di zero = livello logico 1 |

<B1>

| | | | |
|---|---|---|---|
| 311 | C9 | RET | Rientra dalla subroutine |
| 312 | CA | JZ <B2> <B3> | Salta se il flip-flop di zero = livello logico 1 |
| 313 | CB | – | – |
| 314 | CC | CZ <B2> <B3> | Richiama la subroutine se il flip-flop di zero = livello logico 1 |
| 315 | CD | CALL <B2> <B3> | Richiama la subroutine posta a M = <B2> <B3> |
| 316 | CE | ACI <B2> | Somma in modo immediato il flip-flop di carry al registro A |
| 317 | CF | RST 1 | Richiama la subroutine all'indirizzo $010_8$ |
| 320 | D0 | RNC | Rientra dalla subroutine se il flip-flop di riporto = livello logico 0 |
| 321 | D1 | POP D | Estrai dallo stack e memorizza l'indirizzo nella coppia di registri D ed E |
| 322 | D2 | JNC <B2> <B3> | Salta se il flip-flop di carry = livello logico 0 |
| 323 | D3 | OUT <B2> | Poni in uscita verso il dispositivo indirizzato da <B2> |
| 324 | D4 | CNC <B2> <B3> | Richiama la subroutine se il flip-flop di carry = livello logico 0 |
| 325 | D5 | PUSH D | Inserisci i contenuti della coppia di registri D ed E nello stack |
| 326 | D6 | SUI <B2> | Sottrai in modo immediato dal registro A |
| 327 | D7 | RST 2 | Richiama la subroutine all'indirizzo $020_8$ |
| 330 | D8 | RC | Rientra dalla subroutine se il flip-flop di carry = livello logico 1 |
| 331 | D9 | – | – |
| 332 | DA | JC <B2> <B3> | Salta se il flip-flop di carry = livello logico 1 |
| 333 | DB | IN <B2> | Poni in ingresso dal dispositivo indirizzato da <B2> |
| 334 | DC | CC <B2> <B3> | Richiama la subroutine se il flip-flop di carry = livello logico 1 |
| 335 | DD | – | – |
| 336 | DE | SBI <B2> | Sottrai in modo immediato |
| 337 | DF | RST 3 | Richiama la subroutine all'indirizzo $030_8$ |
| 340 | E0 | RPO | Rientra dalla subroutine se il flip-flop di parità = livello logico 0 |
| 341 | E1 | POP H | Estrai dallo stack e momorizza l'indirizzo nella coppia di registri H ed L |
| 342 | E2 | JPO <B2> <B3> | Salta se il flip-flop di parità = livello logico 0 |
| 343 | E3 | XTHL | Scambia la parte superiore dello stack con i contenuti di H ed L |
| 344 | E4 | CPO <B2> <B3> | Richiama la subroutine se il flip-flop di parità è a livello logico 0 |
| 345 | E5 | PUSH H | Inserisci i contenuti della coppia di registri H ed L nello stack |
| 346 | E6 | ANI <B2> | AND in modo immediato con i contenuti del registro A |
| 347 | E7 | RST 4 | Richiama la subroutine all'indirizzo $040_8$ |
| 350 | E8 | RPE | Rientra dalla subroutine se il flip-flop di parità = livello logico 1 |

<B1>

| | | | |
|---|---|---|---|
| 351 | E9 | PCHL | Salto indiretto ad M indirizzato dalla coppia di registri H ed L |
| 252 | EA | JPE <B2> <B3> | Salta se il flip-flop di parità = livello logico 1 |
| 353 | EB | XCHG | Scambia i contenuti dei registri H, L con i registri D, E |
| 354 | EC | CPE <B2> <B3> | Richiama la subroutine se il flip-flop di parità = livello logico 1 |
| 355 | ED | – | – |
| 356 | EE | XRI <B2> | Esegui in modo immediato un'operazione di OR esclusivo con i contenuti del reg. A |
| 357 | EF | RST 5 | Richiama la subroutine all'indirizzo $050_8$ |
| 360 | F0 | RP | Rientra nella subroutine se il flip-flop di segno = livello logico 0 |
| 361 | F1 | POP PSW | Estrai dallo stack memorizza nel registro A e nel PSW |
| 362 | F2 | JP <B2> <B3> | Salta, se il flip-flop di segno = livello logico 0 (segno positivo) |
| 363 | F3 | DI | Disabilita l'interruzione |
| 364 | F4 | CP <B2> <B3> | Richiama la subroutine se il flip-flop di segno = livello logico 0 |
| 365 | F5 | PUSH PSW | Inserisci i contenuti del registro A e i flag sullo stack |
| 366 | F6 | ORI <B2> | OR in modo immediato con i contenuti del registro A |
| 367 | F7 | RST 6 | Richiama la subroutine all'indirizzo $060_8$ |
| 370 | F8 | RM | Rientra dalla subroutine se il flip-flop di segno = livello logico 1 |
| 371 | F9 | SPHL | Trasferisci i contenuti dei registri H,L nello stack pointer |
| 372 | FA | JM <B2> <B3> | Salta, se il flip-flop di segno = livello logico 1 (segno meno) |
| 373 | FB | EI | Abilita l'interruzone |
| 374 | FC | CM <B2> <B3> | Richiama la subroutine se il flip-flop di segno = livello logico 1 |
| 375 | FD | – | – |
| 376 | FE | CPI <B2> | Confronta in modo immediato con i contenuti del registro A |
| 377 | FF | RST 7 | Richiama la subroutine all'indirizzo $070_8$ |

## SOMMARIO DEL SET DI ISTRUZIONI 8080

### ISTRUZIONI A UN BYTE

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| INR r | 0*S*4 | INX B | 003 | POP B | 301 | RNZ | 300 | XCHG | 353 |
| DCR r | 0*S*5 | INX D | 023 | POP D | 321 | RZ | 310 | XTHL | 343 |
| | | INX H | 043 | POP H | 341 | RNC | 320 | SPHL | 371 |
| MOV $r_1r_2$ | 1*DS* | INX SP | 063 | POP PSW | 361 | RC | 330 | PCHL | 351 |
| | | | | | | RPO | 340 | HLT | 166 |
| ADD r | 20*S* | DCX B | 013 | PUSH B | 305 | RPE | 350 | NOP | 000 |
| ADC r | 21*S* | DCX D | 033 | PUSH D | 325 | RP | 360 | DI | 363 |
| SUB r | 22*S* | DCX H | 053 | PUSH H | 345 | RM | 370 | EI | 373 |
| SBB r | 23*S* | DCX SP | 073 | PUSH PSW | 365 | RET | 311 | | |
| ANA r | 24*S* | | | | | | | DAA | 047 |
| XRA r | 25*S* | DAD B | 011 | STAX B | 002 | RLC | 007 | CMA | 057 |
| ORA r | 26*S* | DAD D | 031 | STAX D | 022 | RRC | 017 | STC | 067 |
| CMP r | 27*S* | DAD H | 051 | LDAX B | 012 | RAL | 027 | CMC | 077 |
| | | DAD SP | 071 | LDAX D | 032 | RAR | 037 | RST | 3A7 |

*S e D:* B = 0, C = 1, D = 2, E = 3, H = 4, L = 5, M = 6, accumulatore = 7
*A:* 0 fino a 7

### ISTRUZIONI A DUE BYTE

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ADI | <B2> | 306 | IN | <B2> | 333 | MVI B | <B2> | 006 |
| ACI | <B2> | 316 | OUT | <B2> | 323 | MVI C | <B2> | 016 |
| SUI | <B2> | 326 | | | | MVI D | <B2> | 026 |
| SBI | <B2> | 336 | | | | MVI E | <B2> | 036 |
| ANI | <B2> | 346 | | | | MVI H | <B2> | 046 |
| XRI | <B2> | 356 | | | | MVI L | <B2> | 056 |
| ORI | <B2> | 366 | | | | MVI M | <B2> | 066 |
| CPI | <B2> | 376 | | | | MVI A | <B2> | 076 |

### ISTRUZIONI A TRE BYTE

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| JNZ | <B2> | <B3> | 302 | CNZ | <B2> | <B3> | 304 | LXI B | <B2> | <B3> | 001 |
| JZ | <B2> | <B3> | 312 | CZ | <B2> | <B3> | 314 | LXI D | <B2> | <B3> | 021 |
| JNC | <B2> | <B3> | 322 | CNC | <B2> | <B3> | 324 | LXI H | <B2> | <B3> | 041 |
| JC | <B2> | <B3> | 332 | CC | <B2> | <B3> | 334 | LXI SP | <B2> | <B3> | 061 |
| JPO | <B2> | <B3> | 342 | CPO | <B2> | <B3> | 344 | | | | |
| JPE | <B2> | <B3> | 352 | CPE | <B2> | <B3> | 354 | STA | <B2> | <B3> | 062 |
| JP | <B2> | <B3> | 362 | CP | <B2> | <B3> | 364 | LDA | <B2> | <B3> | 072 |
| JM | <B2> | <B3> | 372 | CM | <B2> | <B3> | 374 | SHLD | <B2> | <B3> | 042 |
| JMP | <B2> | <B3> | 303 | CALL | <B2> | <B3> | 315 | LHLD | <B2> | <B3> | 052 |

## DOMANDE RIEPILOGATIVE

Le seguenti domande vi aiuteranno a ripassare il set di istruzioni 8080.

1. Quali flag vengono coinvolti dalle seguenti istruzioni nel corso della loro esecuzione? Usate le abbreviazioni: Z = flag di zero; C = flag di carry; P = flag di parità; S = flag di segno e AC = flag di carry ausiliario.

   a. JMP

   b. POP B

   c. INX D

   d. STAX D

   e. RST n

   f. LXI SP

   g. RET

   h. SHLD

   i. LHLD

   j. DAA

   k. EI

   l. XTHL

   m. PCHL

   n. DAD B

   o. CMC

   p. CMP r

2. Lo stack pointer è inizialmente HI = 004 e LO = 000. Richiamate una subroutine e poi eseguite le seguenti istruzioni nell'ordine dato:

```
PUSH D
PUSH H
PUSH PSW
PUSH B
```

In quale locazione di memoria nello stack appaiono i contenuti dei registri interni? Rispondete a questa domanda per ogni registro.

3. Quali istruzioni usate per controllare la locazione dello stack?

4. Spiegate le analogie e/o le differenze fra le seguenti coppie di termini. Questa è una domanda di riepilogo che contiene materiale anche di altri capitoli.

a. Registro/coppia di registri

b. Byte/bit

c. Byte/parola

d. Parola/indirizzo di memoria

e. Byte d'indirizzo HI/byte d'indirizzo LO

f. Salto/richiamo

g. Istruzione condizionata/incondizionata

h. OR/OR esclusivo per un gate a 2 ingressi

i. Flag di zero/flag di segno

j. Flag di carry/flag di carry ausiliario

k. Push/Pop

l. Accumulatore/ALU

m. Byte di dati/byte d'indirizzo

n. Code ottale/codice esadecimale

o. Incremento/decremento

p. Incremento/ADD

q. Istruzioni IN/OUT

r. Istruzioni ADD/ADC

s. Istruzioni MOV/MVI

t. Istruzioni MVI/LXI

u. Istruzioni EI/DI

v. Istruzioni SUB A/XRA A

w. Carry/carry negativo

x. Codice macchina/codice mnemonico

y. Coppia di registri B/coppia di registri H

z. Registro istruzioni/decodificatore di istruzioni

**RISPOSTE**

1; a. Nessuno
   b. Nessuno
   c. Nessuno (questo è molto importante)
   d. Nessuno
   e. Nessuno
   f. Nessuno
   g. Nessuno
   h. Nessuno
   i. Nessuno
   j. Coinvolti tutti i flag
   k. Nessuno
   l. Nessuno
   m. Nessuno
   n. Solo il flag di carry
   o. Solo il flag di carry
   p. Coinvolti tutti i flag

2.

| Byte d'indirizzo LO | Contenuti (registro) | |
|---|---|---|
| 377 | byte HI | (dal contatore di programma) |
| 376 | byte LO | |
| 375 | D | |
| 374 | E | |
| 373 | H | |
| 372 | L | |
| 371 | Accumulatore | |
| 370 | Flag | |
| 367 | B | |
| 366 | C | |

3. LXI SP (la più utile)
   INX SP
   DCX SP
   SPHL

Oltre a queste, tutte le istruzioni di rientro dalla subroutine, di richiamo della subroutine, POP e PUSH, influenzano la posizione dello stack.

4. a. Nel chip 8080A, un registro è uno dei registri universali (8 bit) o l'accumulatore. Una coppia di registri è un registro a 16 bit considerato come un'unità e consiste di due registri universali, come B e C, o D ed E.

   b. Un byte è formato da otto bit. Un bit è una decisione binaria singola.

   c. Un byte è una sequenza di cifre binarie adiacenti che operano su di un'unità, ma di solito più breve di una parola macchina, che può consistere di due o più byte.

   d. Un indirizzo di memoria è una sequenza di cifre binarie adiacenti trattate come un'unità, e può rappresentare dati, istruzioni, o altre quantità binarie oltre agli indirizzi di memoria.

   e. In un microcomputer 8080A, il byte d'indirizzo HI è il gruppo di otto bit più significativi nella parola dell'indirizzo di memoria a 16 bit d'indirizzo; LO è il gruppo di otto bit meno significativi.

   f. Entrambe sono istruzioni di salto. Comunque, in un'istruzione di richiamo, i contenuti del contatore di programma vengono mantenuti prima che l'istruzione sia eseguita. In un'istruzione di salto, il contatore di programma è ignorato.

g. Si riferiscono entrambi ad istruzioni di salto nel set di istruzioni 8080A. In un'istruzione di salto condizionato, il fatto che il salto avvenga o meno dipende dallo stato logico del flag selezionato. Un'istruzione di salto incondizionato ignora lo stato logico di tutti i flag.
h. Per un gate OR, quando entrambi gli ingressi sono a livello logico 1, anche l'uscita è a livello logico 1. Per un gate OR esclusivo, quando entrambi gli ingressi sono a livello logico 1, anche l'uscita è a livello logico 0. Sotto altri punti di vista, i due gate sono uguali nelle loro caratteristiche logiche.
i. Il flag di zero è settato solo quando il risultato di un'istruzione aritmetico/logica è zero. Il flag di segno si riferisce allo stato logico del bit più significativo del risultato, non a tutta la parola o al byte. I flag si riferiscono ad altre cose.
j. Il flag di carry si riferisce ad un riporto del bit più significativo in un risultatato a 8 bit nel microprocessore 8080A. Il flag di carry ausiliario si riferisce ad un riporto del bit D3 (il quarto bit) del risultato.
k. Un'istruzione PUSH addiziona due byte allo stack e decrementa lo stack pointer. Un'istruzione POP sposta due byte dallo stack ed incrementa lo stack pointer.
l. L'accumulatore è un singolo registro nell'unità logica-aritmetica (ALU), che contiene altri circuiti digitali richiesti per eseguire operazioni aritmetiche e logiche.
m. Il byte di dati non viene mai caricato nel contatore di programma. Un byte d'indirizzo sí.
n. Codice ottale è un codice a otto stati. Codice esadecimale è un codice binario a sedici stati. I primi otto stati dei due codici sono identici.
o. Incrementare significa aumentare di uno. Decrementare significa diminuire di uno.
p. Incrementare significa aumentare solo di uno. In un'operazione ADD, l'addendo è limitato dal byte o lunghezza di parola; non è limitato all'unità.
q. I dati in ingresso dell'istruzione IN da un dispositivo esterno nell'accumulatore. I dati in uscita dell'istruzione OUT dall'accumulatore ad un dispositivo esterno.
r. L'istruzione ADC è un'istruzione ADD nella quale vengono addizionati anche i contenuti del flag di carry.
s. Il byte che viene trasferito in un'istruzione MVI è contenuto all'interno del programma come il secondo byte dell'istruzione. Il byte che viene trasferito in un'istruzione MOV è originariamente presente in un registro o in una specifica locazione di memoria.
t. L'istruzione MVI trasferisce un byte del programma ad un registro. L'istruzione LXI trasferisce una coppia di byte del programma ad una coppia di registri.
u. L'istruzione EI abilita il flag di interrupt e permette che il chip 8080A venga interrotto. L'istruzione DI disabilita il flag di interrupt e impedisce che il chip 8080A venga interrotto.
v. Entrambe le istruzioni azzerano l'accumulatore e il flag di carry.
w. Si riferiscono entrambi allo stato logico del flag di carry, ma il carry si riferisce allo stato del flag dopo un'operazione di addizione, mentre il carry negativo si riferisce al flag dopo un'operazione di sottrazione.
x. Il codice macchina è rappresentato da cifre binarie, ottali, esadecimali. Il codice mnemonico è più semplice da ricordare, ma deve essere convertito in codice macchina prima di essere eseguito dal microcalcolatore.
y. La coppia di registri H funge da indirizzo del pointer per tutte le istruzioni che si riferiscono direttamente alla locazione di memoria M. Benchè la coppia di registri B possa fungere da indirizzo di memoria, vi sono molte istruzioni ad essa associate quando viene usata come indirizzo del pointer. ADD M, SUB M, INR M, DCR M, XRA M, ORA M, CMP M, ecc. sono alcune delle istruzioni che impiegano la coppia di registri H come indirizzo del pointer.
z. Il registro istruzioni memorizza il codice di operazione a 8 bit in un chip 8080A. Il decodificatore di istruzioni decodifica questa quantità di 8 bit in una serie di azioni.

## INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
   a. Processor (check appropriate box)
   b. Program title: Name or brief description of program function
   c. Function: Detailed description of operations performed by the program
   d. Required hardware:
      For example: TTY on port 0 and 1
      Interrupt circuitry
      I/O Interface
      Machine line and configuration for cross products
   e. Required software:
      For example: TTY routine
      Floating point package
      Support software required for cross products
   f. Input parameters: Description of register values, memory areas or values accepted from input ports
   g. Output results: Values to be expected in registers, memory areas or on output ports
   h. Program details (for resident products only)
      1. Registers modified
      2. RAM required (bytes)
      3. ROM required (bytes)
      4. Maximum subroutine nesting level
   i. Assembler/Compiler Used:
      For example: PL/M
      Intellec 8 Macro Assembler
      IBM 370 Fortran IV
   j. Programmer, company and address

2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps is not necessary.

3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.

---

Send completed documentation to:

**Intel Corporation**
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

CAPITOLO 19

# LE TECNICHE DI BUS DATI NELL'USO DI DISPOSITIVI THREE-STATE

## INTRODUZIONE

Un bus è un insieme di linee di collegamento sulle quali vengono trasferite le informazioni digitali, da una delle molte sorgenti ad una delle molte destinazioni. L'obiettivo fondamentale di un bus è minimizzare il numero di interconnessioni richieste per trasferire le informazioni fra un dispositivo digitale e l'altro. In questo capitolo, descriveremo le tecniche di bus three-state, tecnica che viene attualmente usata nei chip e nei sistemi a microprocessor.

## OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Dare la definizione di "bus" e del relativo verbo (to bus).
- Descrivere le caratteristiche di un buffer three-state, compresi gli ingressi dei dati e di abilitazione/disabilitazione, nonchè l'uscita three-state.
- Scrivere una tabella della verità per un dispositivo three-state.
- Fornire uno o due esempi di semplici sistemi di bus.
- Fare un elenco di cinque-dieci chip a three-state messi a disposizione dalla National Semiconductor Corporation.

## COSA È UN BUS?

Un *bus* digitale è un canale sul quale vengono trasferite le informazioni digitali, da una delle molte sorgenti verso una delle molte destinazioni. Può avvenire solo un trasferimento di informazioni per volta. Mentre tale trasferimento è in corso, tutte le altre sorgenti che sono legate al bus devono essere disabilitate. Il verbo *"to bus"*, significa fare un'interconnessione fra molti dispositivi digitali, che ricevono o trasmettono informazioni digitali, per mezzo di un insieme di canali a conduzione comune, chiamati bus, sui quali vengono trasferite tutte le informazioni digitali fra tali dispositivi.

Lo scopo fondamentale di un bus è miniminizzare il numero di interconnessioni richieste per trasferire le informazioni fra un dispositivo digitale e l'altro. I bus sono presenti all'interno di circuiti integrati, ad esempio il bus di dati interno al chip 8080A; fra un circuito integrato e l'altro, ad esempio i bus dati bidirezionali, bus d'indirizzo e di controllo presenti in un microcomputer 8080A; e fra i sistemi e gli strumenti digitali, ad esempio l'interface bus della Hewlett-Packard che è ora un'interfaccia standard fra gli strumenti digitali.

Sebbene non sia molto discusso nei libri di testo sull'elettronica digitale, il concetto di bus è probabilmente uno dei più importanti concetti dell'elettronica digitale. Senza la possibilità di condividere i canali di informazione, la maggior parte dei dispositivi digitali richiederebbe probabilmente un numero di connessioni di fili tre o quattro volte maggiore di quello attuale. Le piastre di circuito stampato per microcomputer e minicomputer sarebbero notevolmente più complesse..... e costose.

## BUSSING THREE-STATE

In un sistema di bus, il gate ideale dovrebbe avere due stati di uscita digitali (livello logico 0 e livello logico 1) ed un terzo stato non collegato o isolato. Questo caso può essere visto facilmente per mezzo della seguente tabella della verità:

| Dati in ingresso | Segnale di gating | Uscita |
|---|---|---|
| 0 | abilitare | 0 |
| 1 | abilitare | 1 |
| 0 | disabilitare | Scollegata dal bus |
| 1 | disabilitata | Scollegata dal bus |

In altre parole, il terzo stato è una condizione nella quale il gate è "scollegato" dal bus e da questo specifico gate non appare nessun dato in ingresso sul bus.

La soluzione portata avanti dalla National Semiconductor Corporation è l'uscita *TRI-STATE®. È opportuno citare dal loro catalogo, "Circuiti Integrati Digitali", la descrizione del concetto di TRI-STATE:

Caratteristiche

- Serie 54/74 TTL compatibile
- Ad una linea di bus comune si possono collegare fino a 128 buffer
- Ritardo di propagazione di 12 ns
- Capacità di pilotaggio superiore
- Controllo indipendente di ogni buffer

* Nome dato dalla National Semiconductor ai dispositivi THREE-STATE

"La concezione TRI-STATE permette alle uscite di essere unite insieme e poi collegate ad una linea di bus comune. Le normali uscite TTL non possono essere collegate insieme in quanto in uscita sia in logica "1" che 0, presentano bassa impedenza e quindi un dispositivo richiama corrente da un altro. Se comunque sia il transistor di uscita superiore che quello inferiore sono spenti su tutti i dispositivi collegati tranne uno, il dispositivo rimanente nel normale stato di bassa impedenza dovrà fornire o togliere dagli altri dispositivi solo una piccola quantità di corrente di perdita. . . ."

"Un tipico sistema di collegamento è mostrato nella figura . . . . Mentre è vero che in un sistema TTL si potrebbero usare i gate a collettore aperto per eseguire le funzioni logiche di questi elementi tri-state, non si raggiungerebbero però nè l'integrità della forma d'onda nè la velocità ottimale. La bassa impendenza di uscita dei dispositivi tri-state fornisce una buona capacità di pilotaggio e una rapida transizione dal livello logico "0" al livello logico "1", assicurando così sia l'integrità della velocità che della forma d'onda."

"È possibile collegare fino a 128 dispositivi ad una linea di bus comune ed avere ancora una capacità di pilotaggio adeguata per permettere fan-out dal bus."

"Un altro vantaggio di questi buffer è che nello stato di alta impedenza, i loro ingressi non presentano il normale caricamento sui dispositivi pilota. Questo è significativo quando si desidera trasmettere in entrambe le direzioni su di una linea comune."

Riassumendo quanto sopra detto, un dispositivo tri-state ha tre possibili stati di uscita: (1) Uno stato di logica "0", (2) Uno stato di logica "1", e (3) Uno stato di uscita ad alta impedenza che è in effetti disconnesso dalla linea del bus. Tutti i dispositivi a tre stati hanno un pin di entrata chiamato ingresso di *abilitazione/disabilitazione*, che permette ai dispositivi logici o di comportarsi normalmente o di sussistere nello stato di alta impedenza. Quando è abilitato, un dispositivo a tre stati si comporta come un normale dispositivo TTL; quando è disabilitato, si comporta come se, in effetti, non fosse collegato al circuito.

La figura 19-1 seguente mostra la tabella della verità di un tipico dispositivo a tre stati:

| Dati in ingresso | Segnale gating | Dati in uscita |
|---|---|---|
| 0 | abilitare | 0 |
| 1 | abilitare | 1 |
| X | disabilitare | Alta impedenza |

X = irrilevante

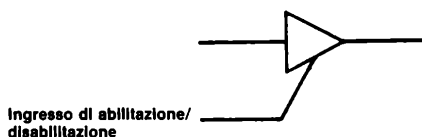



*Figura 19-1.* Schema di un buffer a TRI-STATE che è abilitato da un ingresso logico 1.

## ESEMPI DI SEMPLICI SISTEMI BUS

La Figura 19-2 mostra un semplice sistema bus ad una linea e con quattro dispositivi che è basato sull'uso di un solo chip buffer three-state 74126. Consideriamo il circuito come un sistema bus dal momento che *le uscite dei gate da A a D sono collegate insieme.* Con chip TTL standard serie 7400, non è possibile realizzare questo, a meno che i chip abbiano speciali circuiti di uscita, o a *tre stati* o a *collettore aperto,* che permettano il "bussing".

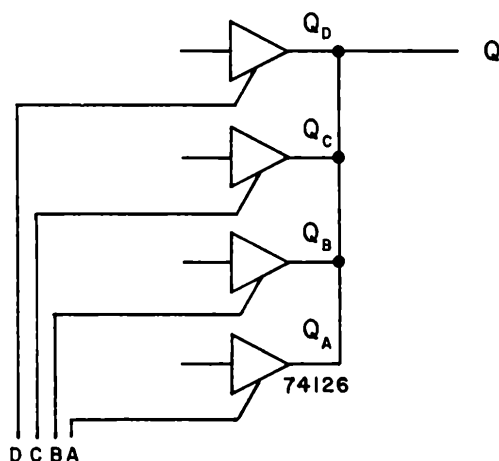



*Figura 19 2.* Un semplice bus ad una linea con quattro dispositivi basato sull'uso di quattro buffer a tre stati 74126.

Se supponiamo che i gate da A a D della figura 19-2 siano abilitati da un ingresso logico 1, il funzionamento del circuito dovrebbe essere chiaro. *Solo un gate del buffer può essere abilitato in qualunque momento; gli altri gate del buffer devono essere disabilitati.* Così, le informazioni digitali provenienti da un solo dei quattro buffer appaiono sul bus ad una sola linea in qualunque momento. Le informazioni degli altri tre buffer vengono bloccate dal momento che i buffer corrispondenti sono disabilitati. Al funzionamento di questo circuito si applica la seguente tabella della verità:

| D | C | B | A | Uscita | Commento |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | $Q_A$ | I buffer B,C,D sono "scollegati" dal bus |
| 0 | 0 | 1 | 0 | $Q_B$ | I buffer A,C,D sono "scollegati" dal bus |
| 0 | 1 | 0 | 0 | $Q_C$ | I buffer A,B,D sono "scollegati" dal bus |
| 1 | 0 | 0 | 0 | $Q_D$ | I buffer A,B,C sono "scollegati" dal bus |

È importante notare che *tutte le altre condizioni di ingresso sono considerate "illegali" per questo circuito, dato che esse permettono che informazioni provenienti da più di un buffer appaiano sul bus ad una sola linea. Inoltre, se tentate di implementare una qualunque di queste condizioni di ingresso "illegali", molto probabilmente farete bruciare il chip three-state!*

I tipici sistemi di bus consistono di bus a più linee, come mostra la Figura 19-3, piuttosto che di bus ad una linea. A parte il fatto che gli ingressi di gating abilitano o disabilitano quattro gate del buffer mostrato nella Figura 19-2. Per esempio, la tabella della verità è essenzialmente la stessa:

| D | C | B | A | Uscita | Commento |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Dispositivo A | I dispositivi B,C,D sono "scollegati" dal bus |
| 0 | 0 | 1 | 0 | Dispositivo B | I dispositivi A,C,D sono "scollegati" dal bus |
| 0 | 1 | 0 | 0 | Dispositivo C | I dispositivi A,B,D sono "scollegati" dal bus |
| 1 | 0 | 0 | 0 | Dispositivo D | I dispositivi A,B,C sono "scollegati" dal bus |

Come nel caso precedente, tutte le altre condizioni di ingresso sono "illegali", dato che permettono che le informazioni provenienti da più di un dispositivo appaiono sul bus.

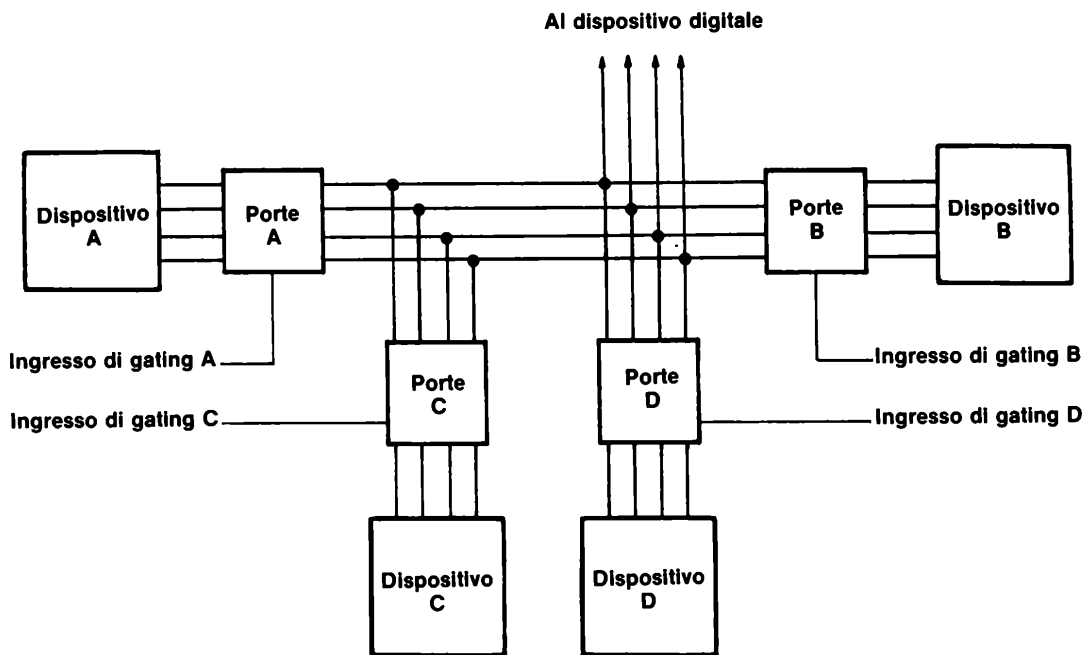
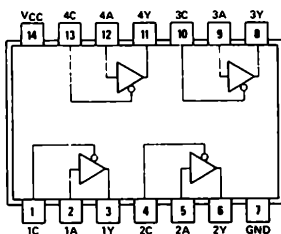



*Figura 19-3.* Un semplice sistema di bus a quattro linee con quattro dispositivi.

## BUFFER THREE-STATE 74125

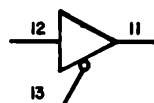
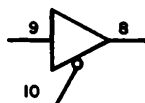
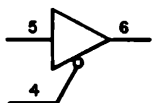
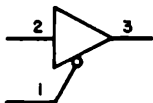
Un tipico buffer three-state 74125 contiene un ingresso di abilitazione/disabilitazione separato in aggiunta ai normali pin di ingresso e uscita,

La figura seguente mostra la configurazione dei pin di un chip 74125, tratta dal "TTL Data Book for Design Engineers" della Texas Instruments:

74125

8093

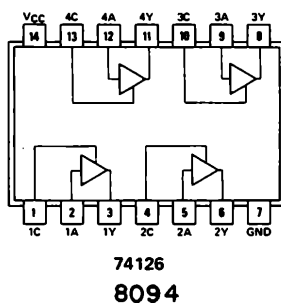
I quattro buffer indipendenti possono essere identificati come segue:

| | |
|---|---|
| Primo buffer: | Ingresso 1A, uscita 1Y, e ingresso di abil./disab. 1C |
| Secondo buffer: | Ingresso 2A, uscita 2Y, e ingresso di abil./disab. 2C |
| Terzo buffer: | Ingresso 3A, uscita 3Y, e ingresso di abil./disab. 3C |
| Quarto buffer: | Ingresso 4A, uscita 4Y, e ingresso di abil./disab. 4C |

Questi quattro buffer possono essere così rappresentati schematicamente

In base alla nostra esperienza, vi raccomandiamo l'uso di questo chip in alternativa al 74126. Quando un ingresso di abilitazione/disabilitazione non è collegato, il buffer corrispondente 74125 è disabilitato.

## BUFFER THREE-STATE 74126

La configurazione dei pin per il buffer three-state 74126 è la seguente:

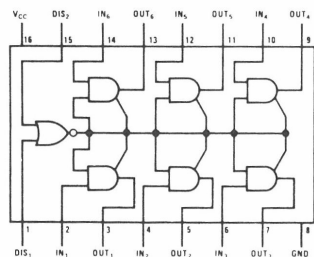
8094

Gli ingressi di alimentazione sono ai pin 7 e 14, e ci sono quattro buffer indipendenti sul chip,

| | |
|---|---|
| Primo buffer: | Ingresso 1A, uscita 1Y, e ingresso di abil./disab. 1C |
| Secondo buffer: | Ingresso 2A, uscita 2Y, e ingresso di abil./disab 2C |
| Terzo buffer: | Ingresso 3A, uscita 3Y, e ingresso di abil./disab. 3C |
| Quarto buffer: | Ingresso 4A, uscita 4Y, e ingresso di abil./disab. 4C |

che possono essere rappresentati schematicamente come segue:

## BUFFER THREE-STATE 8095

Il chip del buffer three-state 8095 contiene sei buffer che sono abilitati simultaneamente dall'uscita di un gate NOR a 2 ingressi. La tabella della verità e la configurazione dei pin sono:

| Ingressi di $DIS_1$ | abil./disab. $DIS_2$ | Dati in ingresso | Uscita del buffer | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | X | Alta impedenza | X=irrilevante |
| 1 | 0 | X | Alta impedenza | |
| 1 | 1 | X | Alta impedenza | |

8095

Questo è un ottimo chip che viene usato di frequente nei circuiti di buffer d'ingresso dei microcomputer. Per esempio, un solo chip 8095 fa sì che sei linee d'ingresso separate vengano collegate al chip di un microprocessor 8080A attraverso il bus dati bidirezionale.

## ALTRI DISPOSITIVI THREE-STATE

Attualmente, la tecnologia di bussing dominante, quella che viene usata nella maggior parte dei microprocessori, è a three-state. Una configurazione circuitale molto comune, che si trova all'interno dei microprocessori, è il latch/buffer three-state, che è mostrato nella Figura 19-4. Esso consiste di un latch tipo D 7475 e di un buffer di uscita three-state che richiede come ingresso di abilitazione uno stato logico 1. Ecco la tabella della verità:

| Clock | Abilitazione | Condizioni di uscita |
|---|---|---|
| 0 | 0 | Latch dei dati precedenti; l'uscita a tre stati è disabilitata |
| 0 | 1 | Latch dei dati precedenti; questi ultimi vengono messi in uscita sul bus |
| 1 | 0 | Il latch segue l'ingresso dei dati; l'uscita a tre stati è disabilitata |
| 1 | 1 | Si comporta come un semplice buffer three-state |

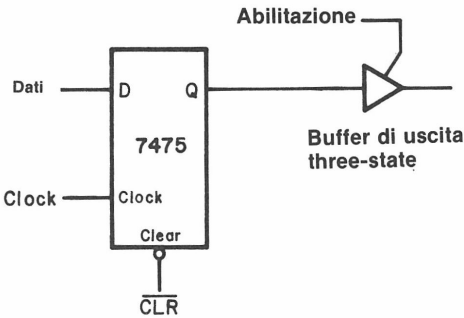



*Figura 19-4.* Circuito latch/buffer three-state che si trova comunemente all'interno dei microprocessori e dei chip di interfaccia associati con i sistemi a microcomputer.

Molti dei nuovi chip di interfaccia programmabili, come l'Intel Corporation 8251, 8253, 8255, 8257 e 8259 impiegano il circuito latch/buffer three-state nei registri interni programmabili a 8 bit.

Sebbene molti chip della serie 7400, compresi il 74125, 74126, 74200, abbiamo uscite three-state, la maggior parte dei dispositivi a tre stati sono messi a disposizione dalla National Semiconductor Corporation, con seconda sorgente la Texas Instruments ed altri. Vi diamo un elenco parziale dei dispositivi "Three-state" che sono disponibili. Da notare che TRI-STATE è un marchio di fabbrica registrato della National Semiconductor.

| | |
|---|---|
| 74200 | Three-state 256-bit read/write memory |
| 74251 | Three-state 8-channel multiplexer |
| 74284 | Three-state 4-bit multiplier |
| 74285 | Three-state 4-bit multiplier |
| 74365 | Three-state hex buffer (simile al 8065) |
| 8093 | Three-state quad buffer (simile al 74125) |
| 8094 | Three-state quad buffer (simile al 74126) |
| 8095 | Three-state hex buffer |
| 8096 | Three-state hex inverter |
| 8097 | Three-sate hex buffer |
| 8098 | Three-state hex inverter |
| 8123 | Three-state quad 2-input multiplexer |
| 8214 | Three-state dual 4:1 multiplexer |
| 8219 | Three-state 16-line-to-1-line multiplexer |
| 8230 | Three-state demultiplexer |
| 8542 | Three-state quad I/O register |
| 8544 | Three-state quad switch debouncer |
| 8551 | Three-state quad D flip-flop |
| 8552 | Three-state decade counter/latch |
| 8553 | Three-state 8-bit latch |
| 8554 | Three-state binary counter/latch |
| 8555 | Three-state programmable decade counter |
| 8556 | Three-state programmable binary counter |
| 8598 | Three-state 256-bit read-only memory |
| 8599 | Three-state 64-bit read/write memory (simile al 74189) |
| 8831 | Three-state line driver |
| 8832 | Three-state line driver |
| 8833 | Three-state quad transceiver |
| 8834 | Three-state quad transceiver |
| 8835 | Three-state quad transceiver |
| 8875 | Three-state 4-bit multiplier |

## INTRODUZIONE AGLI ESPERIMENTI

I seguenti esperimenti dimostrano l'uso delle tecniche di bussing three-state e dei buffer three-state.

| Esperimento N. | Commento |
|---|---|
| 1 | Dimostrare il funzionamento di un solo buffer 74125 con uscita three-state. |
| 2 | Dimostrare come creare un sistema di bus ad una linea con quattro sorgenti usando un solo buffer three-state 74125. |
| 3 | Dimostrare come creare un bus a quattro linee con due sorgenti usando due buffer three-state 74126. Le sorgenti delle informazioni digitali sono un contatore a decade 7490 e un contatore binario 7493. |
| 4 | Dimostrare il funzionamento di un semplice circuito latch/buffer che è basato su di un latch tipo D 7475 e su un buffer three-state 74125. Questo tipo di circuito ad un bit è largamente usato nei registri all'interno dei microprocessori tipo l'8080A. |

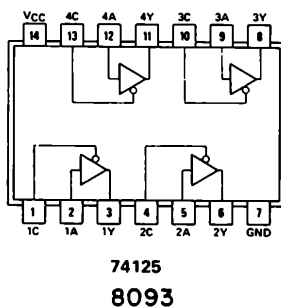
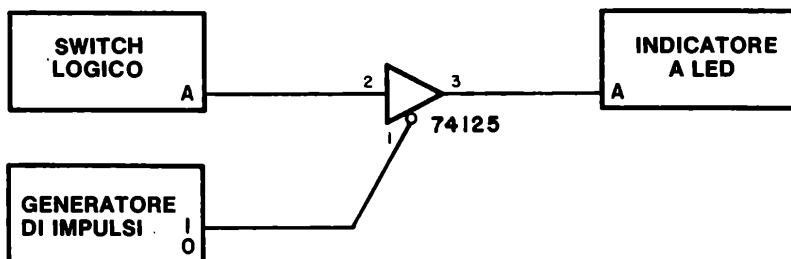
## ESPERIMENTO N. 1

### Scopo

Lo scopo di questo esperimento è di mostrare il funzionamento di un singolo buffer di bus 74125 con uscita three-state.

### Configurazione dei pin del circuito integrato

74125

8093

### Schema del circuito

**Passo 1**
Realizzate il circuito mostrato nella figura. Il chip 74125 contiene quattro buffer di bus indipendenti. Userete solo uno di questi buffer.

**Passo 2**
Posizionate lo switch logico A su uno stato logico 1. Alimentate il breadboard. L'indicatore a LED è acceso o spento?

L'indicatore a LED è spento, il che indica che il buffer è disabilitato o è saltato.

**Passo 3**
Ora premete il tasto del generatore di impulsi. L'indicatore a LED si accende?

Si. Il buffer è ora abilitato con uno stato logico 0.

**Passo 4**
Con il generatore di impulsi premuto, variate lo switch logico fra i livelli logici 1 e 0. Che cosa osservate sull'indicatore a LED?

L'indicatore a LED indica lo stato dello switch logico fino a che il buffer è abilitato.

**Passo 5**
La seguente tabella della verità è quella corretta per il funzionamento del buffer 74125? Se non lo è, scrivete la tabella della verità giusta.

| A | Generatore di impulsi | Indicatore a LED |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

No, la tabella non è corretta. La tabella della verità corretta è:

| A | Generatore di impulsi | Indicatore a LED |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## ESPERIMENTO N. 2

**Scopo**

Lo scopo di questo esperimento è attuare il bussing di quattro diverse sorgenti di dati su di un bus ad una sola linea.

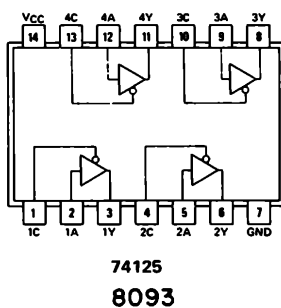
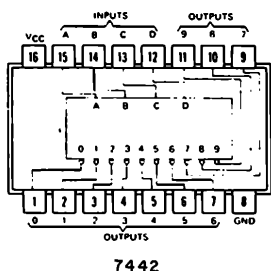
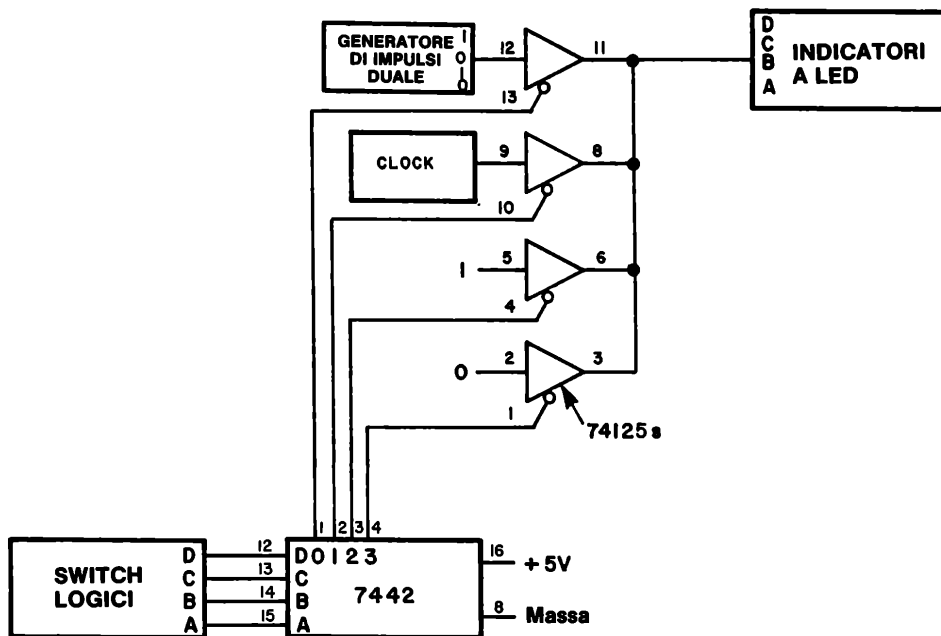
**Configurazioni dei pin dei circuiti integrati**

7442

74125

8093

**Schema del circuito**

GENERATORE DI IMPULSI DUALE
CLOCK
INDICATORI A LED
SWITCH LOGICI
7442
+5V
Massa
74125s

**Passo 1**
Cablate il circuito mostrato nella figura. Qual'è lo scopo del decodificatore 7442?

Lo scopo del chip 7442 nel circuito è quello di abilitare solo un buffer alla volta.

**Passo 2**
Selezionate a turno i canali di uscita 0, 1, 2 e 3 del decodificatore 7442 e scrivete come si presenta l'uscita dell'indicatore a LED nei singoli casi.

Abbiamo osservato i seguenti risultati:

| Canale | Uscita dell'indicatore a LED |
|---|---|
| 0 | Uscita del generatore di impulsi |
| 1 | Uscita del clock |
| 2 | 1 (indicatore acceso) |
| 3 | 0 (indicatore spento) |

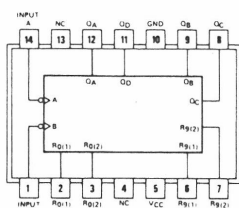
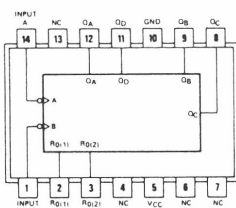
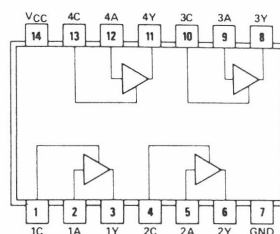
**Passo 3**
Che cosa succede quando scegliete i canali da 4 a 9 sul chip 7442? Non osservate nessuna uscita dell'indicatore a LED?

Osserviamo che l'uscita dell'indicatore a LED è rimasta a livello logico 0. Il motivo è che tutti i buffer 74125 sono disabilitati.

Ricordatevi che in qualunque sistema di bus three-state, un solo ingresso dati del bus deve essere abilitato in qualunque momento. In questo esperimento, il decodificatore 7442 assicura che solo un buffer 74125 alla volta venga abilitato. L'uso, a questo scopo, di decodificatori, è comune nei sistemi di bus three-state.

## ESPERIMENTO N. 3

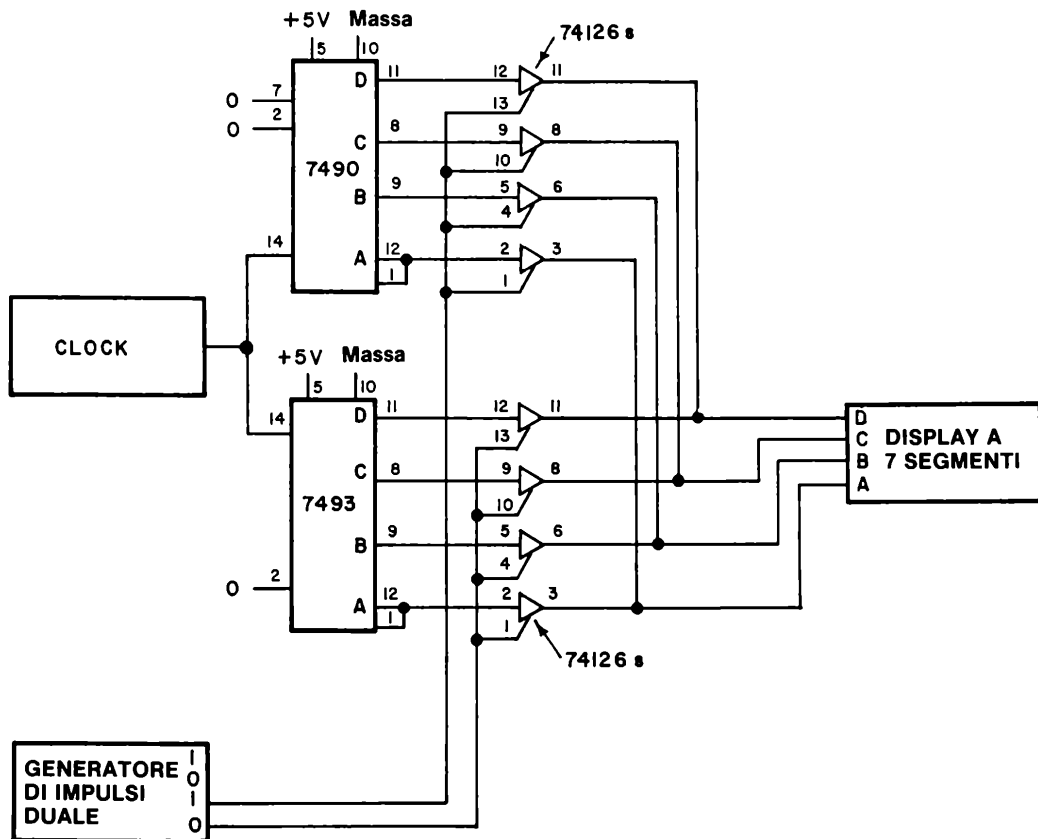
### Scopo
Lo scopo di questo esperimento è attuare il bussing di due dispositivi digitali diversi, un contatore 7490 e un contatore 7493, su di un solo display a LED a sette segmenti usando una coppia di buffer di bus 74126.

### Configurazioni dei pin dei circuiti integrati

### Passo 1
Studiate attentamente lo schema del circuito. Osservate che sono necessari due buffer 74126. Gli ingressi di abilitazione/disabilitazione di ogni chip sono uniti o all'uscita "0" o all'uscita "1" su di un solo generatore di impulsi. Perché viene usato solo un generatore di impulsi?

Per assicurare il fatto che un solo dispositivo di ingresso venga abilitato in qualunque momento.

### Passo 2
Cablate il circuito e poi alimentate il breaboard. Quale contatore viene alimentato, il contatore a decade 7490 o il contatore binario 7493?

È abilitato il contatore a decade 7490, dato che è necessario un ingresso di abilitazione di livello logico 1 per abilitare i buffer 74126.

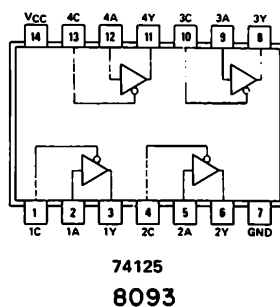
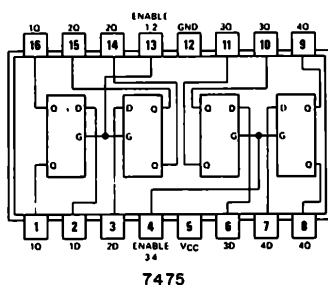
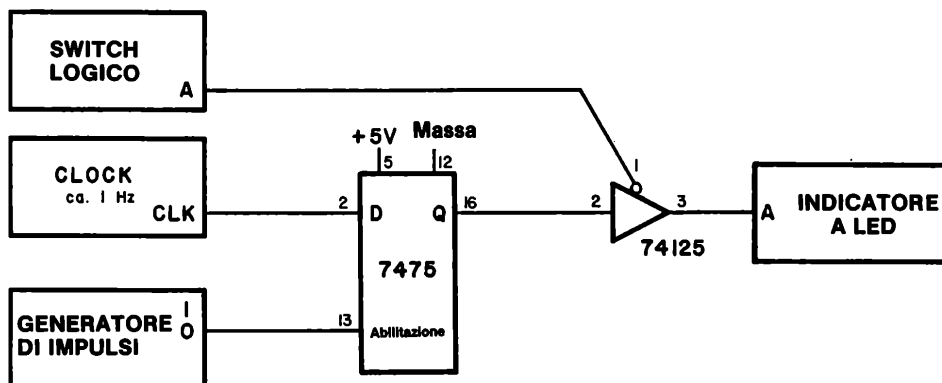
**Schema del circuito**

**Passo 3**
Premete il tasto del generatore di impulsi. Quale contatore viene ora abilitato?

Il contatore binario 7493. Nel suddetto circuito, esiste la possibilità che tutti e due i contatori possano venire abilitati simultaneamente?

No.

## ESPERIMENTO N. 4

**Scopo**
Lo scopo di questo esperimento è provare un semplice circuito latch/buffer basato su di un latch 7475D e su di un buffer three-state 74125.

**Configurazioni dei pin dei circuiti integrati**

**Schema del circuito**

**Passo 1**
Cablate il circuito mostrato nella figura. Sistemate l'uscita clock in maniera che la frequenza di clock sia approssimativamente di 1 Hz.

**Scopo 2**
La tabella della verità per l'operazione del circuito può essere riassunta nel modo seguente:

| Abilitazione 7475 | Disabilitazione 74125 | D | Uscita del buffer three-state |
|---|---|---|---|
| 0 | 0 | 0 | D precedentemente sottoposto a latch |
| 0 | 0 | 1 | D precedentemente sottoposto a latch |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | X | Stato di alta impedenza |
| 1 | 1 | X | Stato di alta impedenza |

X= irrilevante

Quale stato logico disabilita il buffer 74125?

Uno stato logico 1.

**Passo 3**
Abilitate sia il latch 7475 che il buffer 74125. Che cosa osservate sull'indicatore a LED?

Una serie di impulsi di clock che appaiono ad una cadenza di circa 1 Hz.

**Passo 4**
Ora disabiliatte il buffer 74125. Che cosa succede all'uscita dell'indicatore a LED

Si presenta uno stato logico 0. L'uscita del buffer 74125 è ora nel suo stato di alta impedenza.

**Passo 5**
Abilitate il buffer 74125, ma questa volta abilitate o disabilitate il latch 7475. È possibile effettuare un latch dell'ingresso dati dello stato logico 0 o stato logico 1 a D(pin 2)?

Sì, è possibile operare con il latch 7475 indipendentemente dal buffer 74125.

## DOMANDE RIEPILOGATIVE

Le domande seguenti vi aiuteranno a fare un riepilogo dei dispositivi a three-state e delle tecniche di bussing three-state.

1. Che cosa è un bus digitale e perché viene usato?

2. In quali tipi di dispositivi digitali potete trovare un bus?
   Fate un elenco di almeno tre di questi dispositivi.

3. Scrivete una tabella della verità per un semplice dispositivo three-state.

4. Perché un latch/buffer three-state è un circuito tanto utile?

5. Fate un elenco di diversi tipi di dispositivi digitali, ad esempio gate, latch, ecc. che sono messi a disposizione dalla National Semiconductor Corporation con uscite three-state.

6. Che cosa succede in un sistema di bus three-state quando due o più sorgenti di dati del bus vengono abilitate simultaneamente?

**RISPOSTE**

1. Un bus digitale è un insieme di linee di collegamento sulle quali sono trasferite le informazioni digitali. Può avere luogo solo un trasferimento di informazioni alla volta. Mentre tale trasferimento ha luogo, tutte le altre sorgenti legate al bus devono essere disabilitate. Lo scopo fondamentale di un bus è minimizzare il numero di interconnessioni richieste per trasferire le informazioni fra un dispositivo digitale e l'altro.

2. I bus si trovano: (a) all'interno di circuiti integrati quali quello del microprocessor 8080A e delle interfacce programmabili 8251, 8253, 8257, 8259; (b) sulle piastre di circuito stampato che contengono collegamenti di circuiti integrati fra i quali l'informazione deve essere sottoposta a bussing; e (c) negli strumenti digitali, quali minicomputer, microcomputer, frequenzimetri, voltmetri digitali, e simili.

3.

| Dati in ingresso | Segnale gating | Dati in uscita |
|---|---|---|
| 0 | Abilitazione | 0 |
| 1 | Abilitazione | 1 |
| X | Disabilitazione | Alta impedenza |

X=irrilevante

4. Un circuito di questo tipo può funzionare in molti modi diversi: (a) come un latch che memorizza i dati in ingresso ma non li mette in uscita sul bus; (b) come un semplice buffer three-state che non effettua un latch sui dati in ingresso; e (c) come un latch che memorizza i dati in ingresso e li mette in uscita su di un bus.

5. Buffer, latch, contatore, invertitore, multiplexer, memoria di lettura/scrittura, memoria di sola lettura, ricetrasmettitore, demultiplexer, contatore/latch, flip-flop e moltiplicatore.

6. Avvengono due cose: (a) il ricevitore dell'informazione si confonde, poiché non può interpretare l'informazione del bus, e (b) i buffer three-state possono bruciare.

CAPITOLO 20

# INTRODUZIONE ALLE TECNICHE DI I/O TRAMITE L'ACCUMULATORE

## INTRODUZIONE

L'obiettivo di un'operazione I/O (input/output) in un microcomputer è trasferire i dati fra un dispositivo di I/O e uno dei registri interni al chip 8080A. Nell'I/O dell'accumulatore, sono usate le istruzioni IN e OUT, ed il trasferimento dei dati avviene fra l'accumulatore e il dispositivo di I/O. In questo capitolo, imparerete come scrivere semplici programmi di I/O e come cablare semplici circuiti di interfaccia che, lavorando insieme, vi permettono di trasferire i dati di I/O da e verso il chip 8080A.

## OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Stabilire l'obiettivo di un'operazione di I/O di un microcomputer.
- Distinguere fra I/O dell'accumulatore e I/O di memoria nei microcomputer 8080A.
- Delineare molti semplici circuiti di latch che sono utili per l'uscita dell'accumulatore in un microcomputer basato sull'8080A.
- Delineare uno o due semplici circuiti buffer three-state che sono utili per l'ingresso dei dati nell'accumulatore in un microcomputer basato sull'8080A.
- Spiegare il significato della capacità di pilotaggio in uscita nei circuiti di uscita dei microcomputer.
- Spiegare come vengono usati gli impulsi di selezione dispositivo per ottenere l'I/O dell'accumulatore.
- Riassumere le istruzioni di uscita dell'accumulatore nel set di istruzioni 8080A.
- Realizzare un semplice circuito di uscita da un microcomputer.
- Realizzare un semplice circuito di ingresso in un microcomputer.

## CHE COSA È "INPUT/OUTPUT" (INGRESSO/USCITA)?

Quando usiamo il termine *"input/output"* o *I/O,* solitamente intendiamo dire che uno o più byte di dati vengono trasferiti fra un dispositivo di I/O (Vedi Capitolo 16) e il microprocessor. I concetti importanti associati con questo trasferimento di dati sono riassunti nelle Figure da 20-1 a 20-3.

*L'obiettivo di un'operazione di I/O di un microcomputer basato sull'8080A è trasferire i dati fra un dispositivo di I/O e uno dei registri interni al chip 8080A.* Come mostra la Figura 20-1, i registri disponibili comprendono l'accumulatore e i registri universali B,C,D,E,H,L. Non è possibile caricare o lo stack pointer o i registri del contatore di programma direttamente da un dispositivo di I/O esterno. Se si usano le istruzioni di I/O IN e OUT (Vedi capitolo 17), il trasferimento dei dati avviene fra il dispositivo di I/O esterno e l'accumulatore interno al chip 8080A. Questo tipo di operazione di I/O è chiamata *I/O isolated* dalla Intel Corporation, e *I/O dell'accumulatore* dagli altri. Se le istruzioni di riferimento alla memoria, quali MOV M,r o MOV r,M vengono usate, il trasferimento dei dati avviene fra il dispositivo di I/O esterno e *uno qualunque dei registri universali.* Questo secondo tipo di operazione di I/O è chiamata *I/O di memoria.*

*Tutti gli I/O di un microcomputer avvengono a otto bit alla volta sul bus dati bidirezionali dell'8080A.* Come mostra la Figura 20-2, l'unico canale sul quale i dati possono essere trasferiti nel chip 8080A, è il bus dati bidirezionale, da D0 a D7, che è un bus three-state a 8 bit.

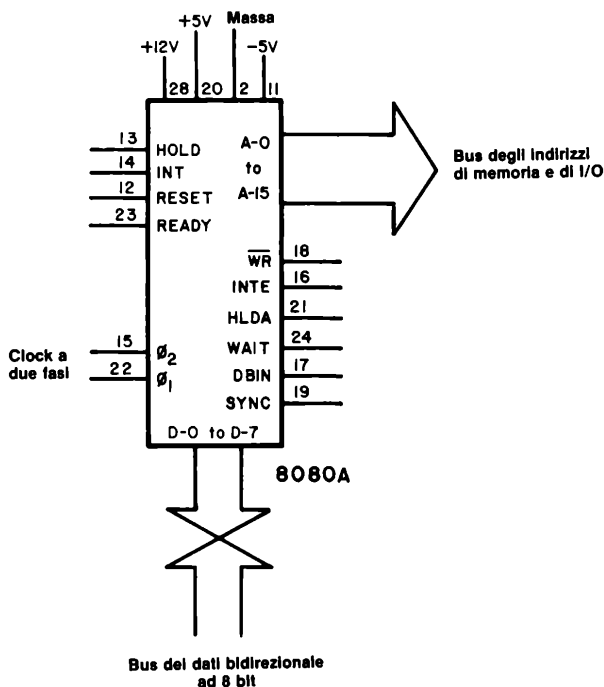



*Figura 20-2.* Schema a blocchi del chip 8080A. I soli pin bidirezionali del chip sono quelli associati con il bus di dati bidirezionale a 8 bit, da D0 a D7.

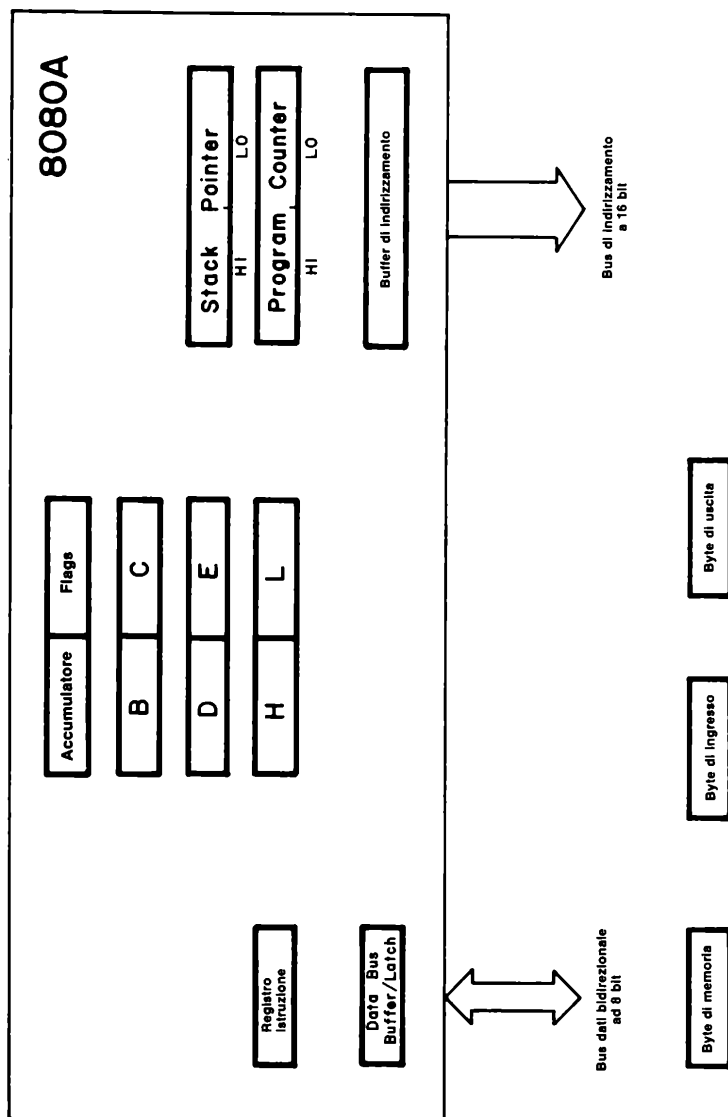



*Figura 20-1A.* Schema dell'archittettura dei registri all'interno del chip 8080A. Il trasferimerimento dei dati fra un dispositivo di I/O e il chip 8080A avviene fra i sette registri interni A,B,C,D,E,H,L, e il dispositivo esterno di I/O. Tutti i trasferimenti di dati avvengono sul bus dati bidirezionale a 8 bit.

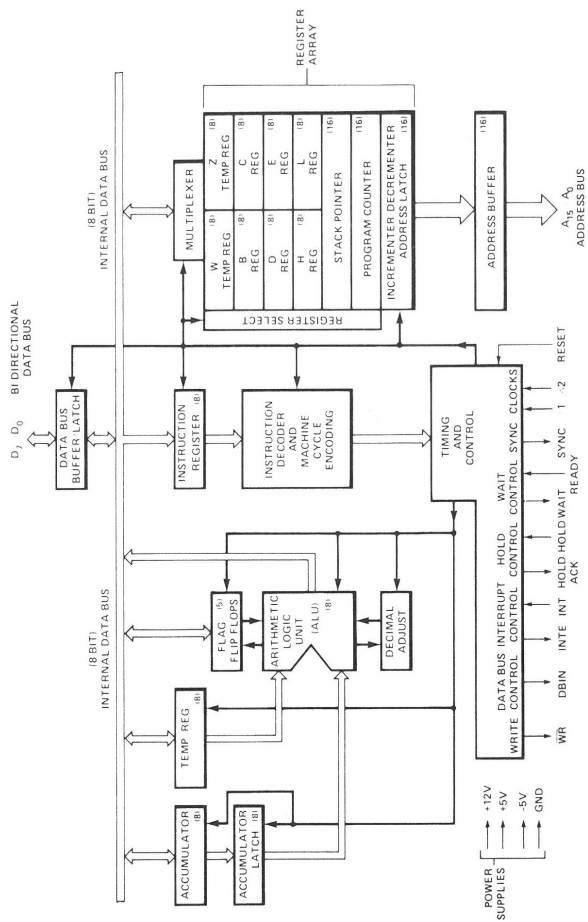



*Figura 20-1B.* Schema più dettagliato dell'architettura dei registri all'interno del chip 8080A. Si possono qui vedere più chiaramente le caratteristiche del bus dati interno a 8 bit.

*Sono necessari impulsi di sincronizzazione per controllare tutti i trasferimenti di dati da e verso il chip 8080A.* Per l'I/O dell'accumulatore, essi vengono chiamati *impulsi di selezione dispositivo* (Vedi capitolo 17), mentre per l'I/O di memoria sono chiamati *impulsi di selezione indirizzo.* Nell'I/O dell'accumulatore, le due istruzioni IN e OUT vi permettono di selezionare e sincronizzare l'operazione di 256 "dispositivi" di ingresso diversi e 256 diversi "dispositivi" di uscita, come mostra la Fig. 20-3. Per un microcomputer basato sull'8080A, tali impulsi hanno la durata di circa un periodo di clock e possono essere impulsi positivi o negativi, a seconda dello schema di decodificazione usato. Tipicamente, tali impulsi sono generati come impulsi di selezione dispositivo negativi da un chip di decodificazione (Vedi Capitolo 17) e devono essere invertiti se vengono richiesti impulsi di selezione dispositivo positivi.

Il punto di riferimento per i termini "input" e "output" è il chip. I dati di uscita dal chip 8080A verso un dispositivo di uscita ("output"), e i dati in ingresso da un dispositivo d'ingresso ("input"). Questo è valido per tutti i casi.

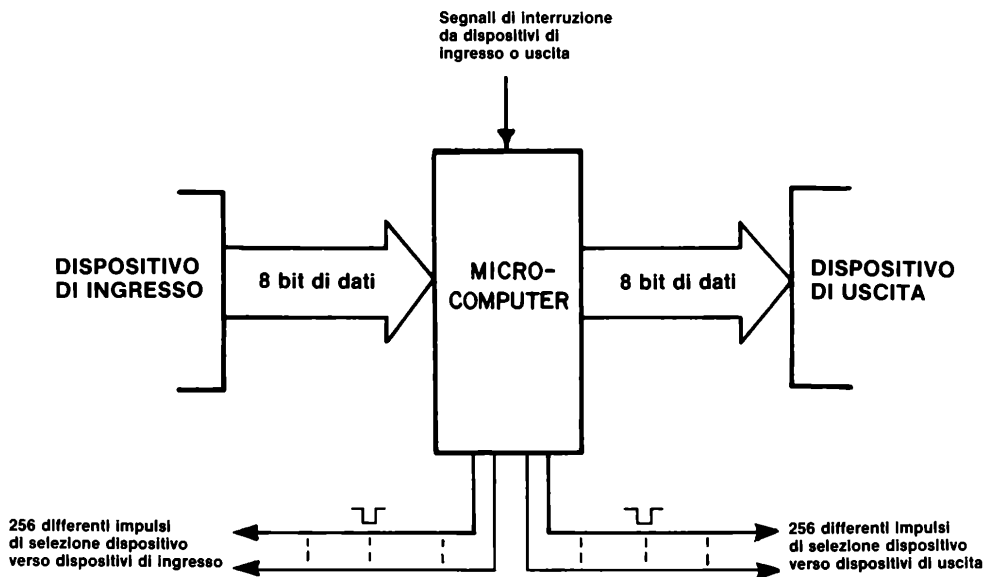



*Figura 20-3.* Schema che illustra il ruolo degli impulsi di selezione dispositivo nell'I/O dell'accumulatore.

## USCITA DEL MICROCOMPUTER

La tecnica di base che si usa per mettere i dati in uscita dall'accumulatore verso un dispositivo di uscita è molto semplice: *si genera, per mezzo del software e dell'hardware, un solo impulso di selezione dispositivo di uscita e lo si usa per abilitare un chip latch nell'istante in cui i dati dell'accumulatore appaiono sul bus di dati bidirezionale.*

Il chip 8080A è responsabile di tutto il processo di sincronizzazione. Il chip di latch ha un ruolo passivo nel trasferimento dei dati ed esegue un latch sui dati stessi solo quando un impulso di selezione dispositivo gli dà un'istruzione del genere. A seconda del tipo di chip latch usato, è impiegato un impulso di selezione positivo o negativo per eseguire un latch sui dati. Richiamatevi all'Esperimento N.4 del capitolo 17, nel quale avete usato un chip decodificatore 74154 per generare sedici diversi impulsi di selezione dispositivo negativi. (Figura 20-4).

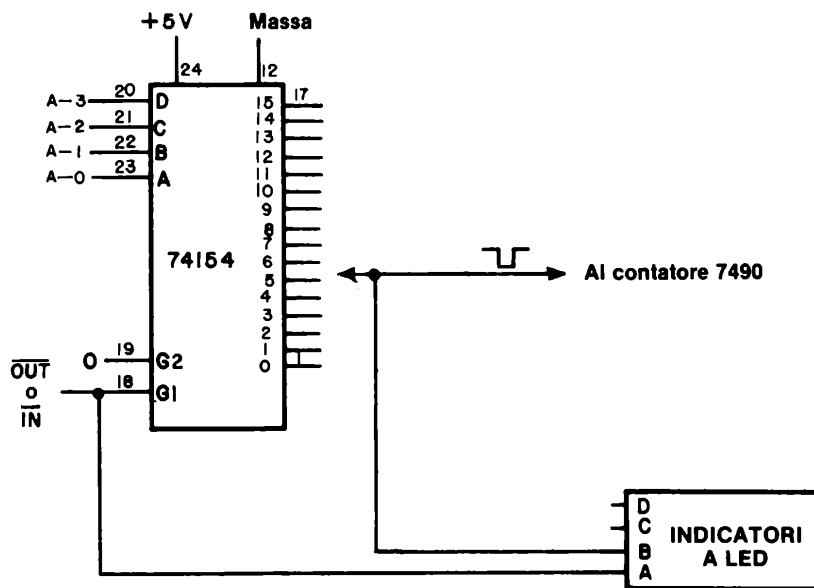



*Figura 20-4.* Schema del circuito che indica l'uso di un chip decodificatore 74154 per generare sedici diversi impulsi di selezione dispositivo negativi.

Sono questi impulsi che vengono usati, o direttamente o con l'inversione, per eseguire un latch dei dati del microcomputer in chip quali l'8212, 74100, 7475, 74198, 74175, o 74193.

## ALCUNI CIRCUITI DI LATCH DI USCITA

I tipici circuiti di uscita del microcomputer comprendono quelli basati sul chip 8212 (Figure 20-5 e 20-6);

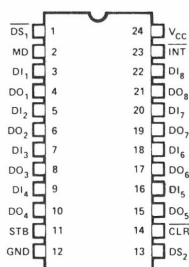



*Figura 20-5.* Configurazione dei pin del latch/buffer a 8 bit 82112.

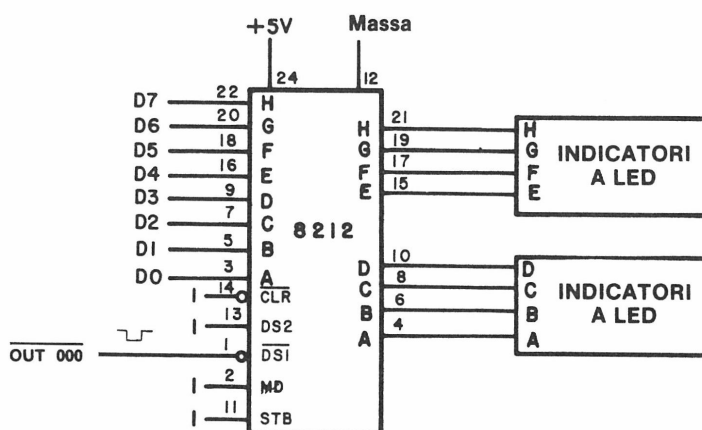



*Figura 20-6.* Schema di un circuito in cui un chip 8212 funge da latch di uscita.

Il latch tipo D a otto bit 74100 (Figure 20-7 e 20-8); una coppia di latch tipo 7475D (Figure 20-9 e 20-10); il registro a scorrimento a otto bit 74198 e il latch a otto bit 74175, entrambi i quali sono dispositivi positive edge triggered (Figure 20-11 e 20-12) e una coppia di contatori up/down 74193 (Figure 20-13 e 20-14).

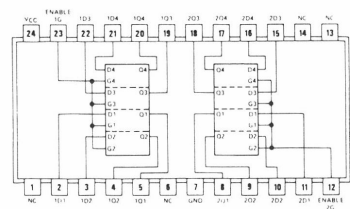



*Figura 20-7.* Configurazione dei pin del latch tipo D a otto bit 74100.

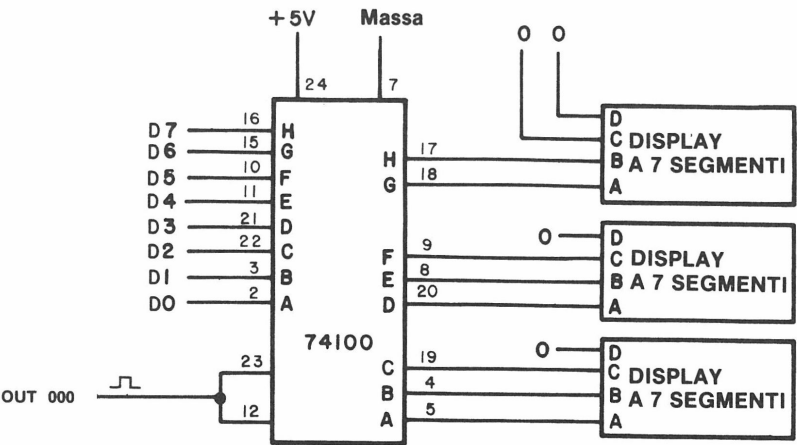



*Figura 20-8.* Circuito di latch di uscita dal microcomputer basato sull'uso di un latch tipo D 74100. L'uscita è fornita come una parola ottale a tre cifre.

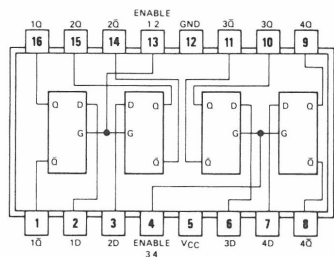



*Figura 20-9.* Configurazione dei pin del latch tipo D a quattro bit 7475.

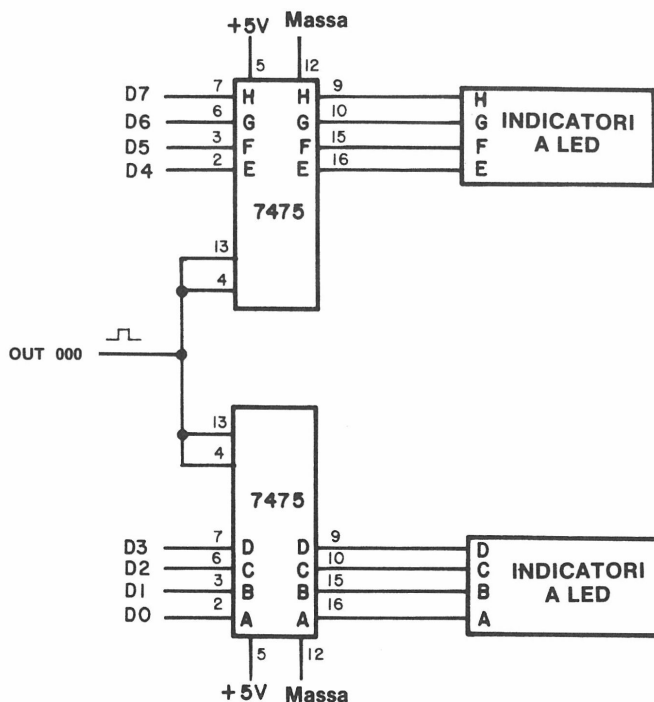



*Figura 20-10.* Circuito di latch di uscita dal microcomputer basato sull'uso di una coppia di latch tipo D a quattro bit 7475.

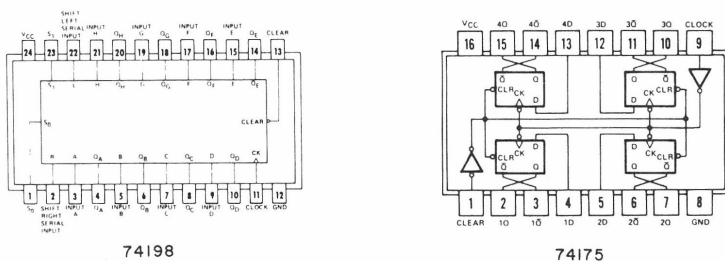



*Figura 20-11.* Configurazioni dei pin del registro a scorrimento a otto bit 74198 e del latch a quattro bit 74175. Entrambi i chip contengono flip-flop positive edge triggered del tipo 7474.

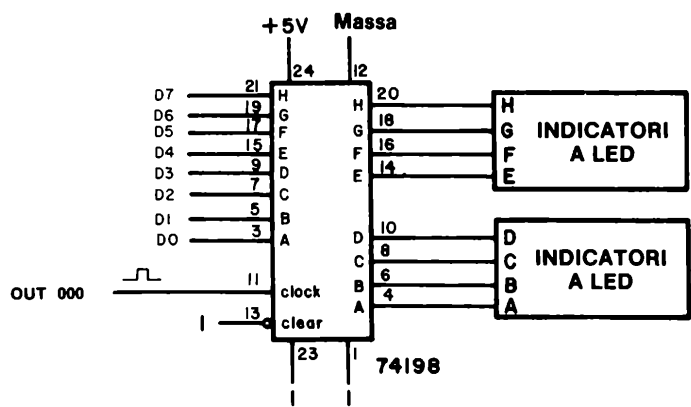



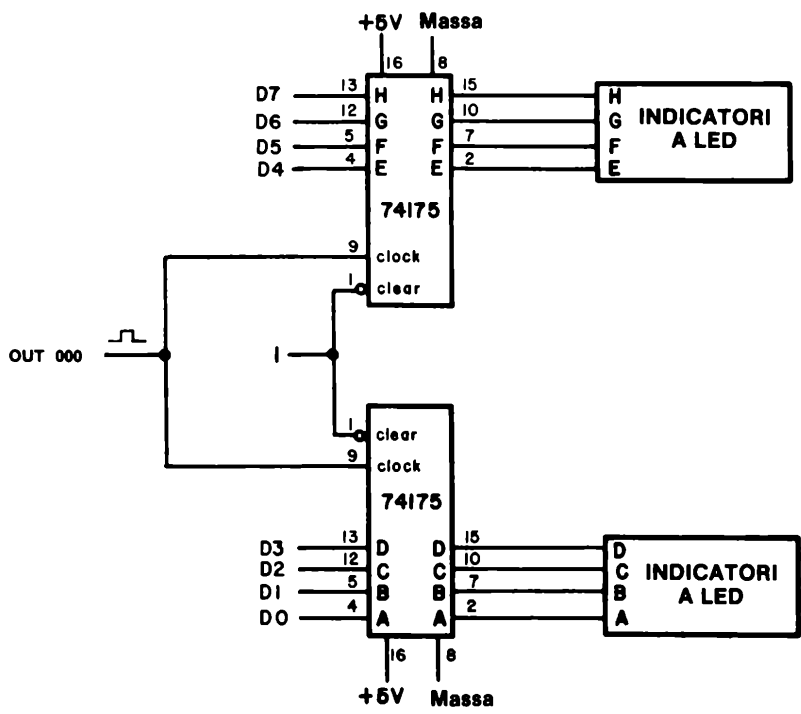
*Figura 20-12.* Circuito di latch di uscita dal microcomputer, basato sull'uso di un registro a scorrimento a otto bit 74198.



*Figura 20-13.* Circuito di latch di uscita dal microcomputer, basato sull'uso di una coppia di latch 74175.

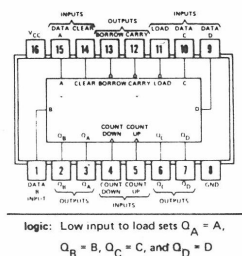



*Figura 20-14.* Configurazione dei pin dei contatori up/down 74192 e 74193; contengono ognuno un latch interno a 4 bit del tipo 7475.

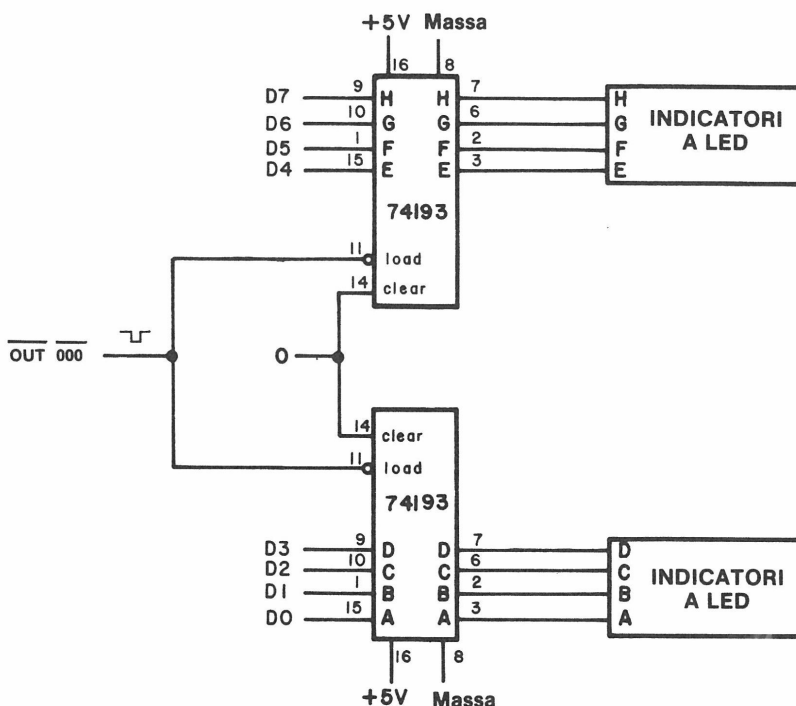



*Figura 20-15.* Circuito di latch di uscita dal microcomputer basato sull'uso di una coppia di contatori up/down 74193. Questo circuito dimostra come si possa precaricare un conteggio in un contatore up/down direttamente dall'accumulatore in un microcomputer 8080A. Un chip di questo tipo non verrebbe usato normalmente come latch universale.

Quando usate il chip 8212 come latch di uscita, dovreste accertarvi che l'ingresso di azzeramento, $\overline{CLR}$, al pin 14, sia collegato al livello logico 1 quando non è usato.

## CAPACITA' DI PILOTAGGIO IN USCITA

Le caratteristiche delle diverse sottofamiglie TTL e i concetti di "fan-in e "fan-out" sono già stati descritti nel capitolo 10. Tali considerazioni sono molto importanti quando costruite circuiti di uscita da un microcomputer. Il fan-out di un microprocessor 8080A è 1,2, il che significa che un pin di uscita può condurre una corrente massima di solo 1,9 mA. Il fan-in di un normale ingresso della serie 7400 è 1,6 mA, perciò dovrebbe essere chiaro che dovreste *sempre usare dispositivi di uscita che abbiano un fan-in più basso in qualunque momento facciate un collegamento diretto con un pin di uscita dall'8080A.*

Quali chip dovreste usare? Vi raccomandiamo i seguenti:

- Chip della sottofamiglia 74LS, nei quali il fan-in di un ingresso è solo 0,2, o 0,32 mA.
- Chip della sottofamiglia 74L, nei quali il fan-in di un ingresso è solo 0,1, o 0,16 mA.
- Chip compatibili con i microprocessori, quali il decodificatore 8205, la porta I/O a otto bit 8212, la memoria statica di lettura/scrittura 8111-2, e i chip relativi, nei quali il fan-in è solo 0,15, o 0,25 mA. Tali chip vengono costruiti specificatamente per interfacciare un 8080A.

Molti chip a bassa potenza possono essere legati ai bus di uscita di un microprocessor, ammesso che tali bus non siano sovraccarichi.

È importante considerare anche il fan-out di un chip a bassa potenza collegato ad un bus dell'8080A. Se i segnali di uscita devono viaggiare su una distanza maggiore di 10 cm, è bene mettere un buffer alle uscite del latch. Le uscite del latch e del flip-flop sono sensibili ai problemi di pilotaggio. Il fan-out di un'uscita 74LS è solo 5, mentre quello di un'uscita 74L è 2,25.

Nel costruire un microcomputer, si hanno comunemente dei bus che sono lunghi da 25 a 30 cm. Non fate un bus che sia più lungo di 30 cm senza usare speciali bus driver.

## INGRESSO AL MICROCOMPUTER

La tecnica che si usa per inserire i dati da un dispositivo esterno nell'accumulatore è analoga alla tecnica usata per l'uscita dal microcomputer: *per mezzo del software e dell'hardware, si genera un solo impulso di selezione dispositivo e lo si usa per abilitare un buffer three-state nell'istante in cui viene aperto un canale diretto fra il bus bidirezionale e l'accumulatore.* Come nell'uscita del microcomputer, il chip 8080A è responsabile di tutto il processo di sincronizzazione. Il chip del buffer three-state ha un ruolo passivo nel processo di trasferimento dati ed applica i dati solo al bus dati quando un impulso di selezione dispositivo gli dà un'istruzione di questo genere. Per abiliatre il buffer si usa un impulso di selezione dispositivo negativo o positivo, a seconda del tipo di buffer usato. Tipici buffer three-state sono l'8212 e l'8095. Il chip di interfaccia programmabile 8255 è diventato famoso come buffer d'ingresso.

## ALCUNI CIRCUITI D'INGRESSO BASATI SUL BUFFER THREE-STATE

I tipici circuiti d'ingresso al microcomputer, comprendono quelli basati sui chip 8095 o 8212, come mostrano le Figure 20-16, 20-17, 20-18. Il latch/buffer a otto bit 8212 è già stato mostrato nella Figura 20-5 come latch di uscita. Andrebbe sottolineato il fatto che, *in un qualunque momento, si deve abilitare solo un ingresso del buffer three-state del microcomputer 8080A.* Tutti gli impulsi di selezione dispositivo d'ingresso dovrebbero essere decodificati in assoluto, il che significa che, per l'I/O dell'accumulatore, tutti gli otto bit del codice dispositivo dovrebbero essere usati per identificare unicamente il dispositivo d'ingresso desiderato.

Nella Figura 20-17, i buffer three-state abilitati fanni sì che i dati vengano trasferiti sulle linee del bus dati bidirezionale, da D0 a D7, che sono collegate alle uscite dei chip 8095. L'accumulatore acquisisce i dati dallo switch logico durante il periodo di clock dell'impulso di selezione dispositivo d'ingresso che, per il chip 8095, è un impulso negativo. Gli ingressi dei chip 8095 possono essere collegati a qualunque sorgente di dati digitali, come uno strumento di laboratorio. Questi dati vengono trasferiti per mezzo dei buffer 8095, posti sulle linee del bus di dati bidirezionale, e *copiati* nell'accumulatore durante un'istruzione IN del microcomputer. I dati vengono inseriti nell'accumulatore ogni volta che vengono eseguiti l'istruzione IN e il codice dispositivo. L'accumulatore non ha bisogno di essere azzerato prima dell'istruzione IN, dato che sia un livello logico 0 che un livello logico 1 vengono spinti nelle posizioni di bit appropriate durante il periodo di clock dell'impulso di selezione dispositivo.

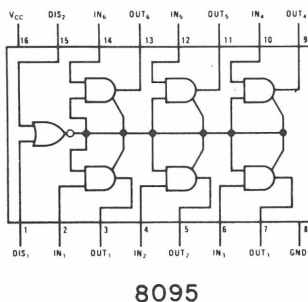



*Figura 20-16.* Configurazione dei pin del buffer three-state 8095: economico e viene largamente usato nei circuiti di interfaccia dei microcomputer.

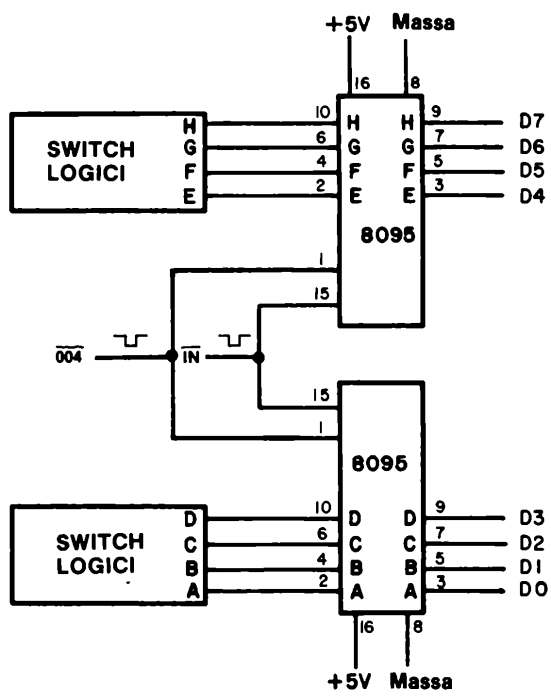



*Figura 20-17.* Circuito d'ingresso al microcomputer basato sull'uso di una coppia di buffer three-state 8095. Gli switch logici possono essere sostituiti da qualunque sorgente a otto bit di dati TTL.

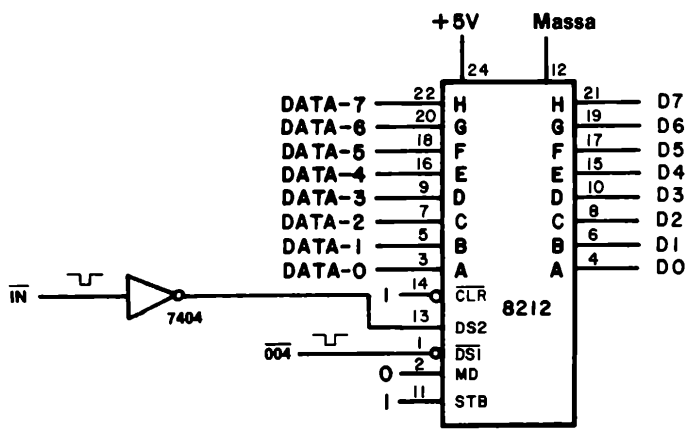



*Figura 20-18.* Circuito d'ingresso al microcomputer, basato sull'uso di un latch/buffer 8212.

## ISTRUZIONI DI I/O DELL'ACCUMULATORE

Vi sono solo due istruzioni di I/O dell'accumulatore 8080A che trasferiscono i dati fra l'accumulatore e i dispositivi di I/O esterni aggiungendosi alla generazione degli impulsi di sincronizzazione $\overline{IN}$ e $\overline{OUT}$:

| | | | |
|---|---|---|---|
| 323 | <B2> | OUT | Mette in uscita i contenuti dell'accumulatore sul latch di uscita selezionato dal codice dispositivo nel secondo byte. Questa istruzione viene eseguita in dieci cicli di clock, o 13,33 μs per un microcomputer 8080A che opera a 750 kHz. |
| 333 | <B2> | IN | Inserisce nell'accumulatore i contenuti del dispositivo digitale e del circuito del buffer a tre stati selezionato dal codice dispositivo nel secondo byte. Questa istruzione viene eseguita in dieci cicli clock, o 13,33 μs per una frequenza di 750 kHz. |

In questo capitolo, al contrario del capitolo N. 17, gli impulsi di selezione dispositivo generati dalle suddette istruzioni vengono usati per trasferire le informazioni da e verso l'accumulatore:

## PRIMO PROGRAMMA DI I/O

Vi mostriamo un semplice programma per inserire i dati dello switch logico della Figura 20-17 nell'accumulatore e poi metterli immediatamente in uscita sul latch di uscita 000 mostrato nella Figura 20-10.

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci dati dallo switch logico associato al dispostivo d'ingresso 004, una coppia di buffer three-state 8095. |
| 001 | 004 | 004 | Codice dispositivo 004. |
| 002 | 323 | OUT | Poni in uscita i dati dell'accumulatore sul latch di uscita 000, coppia di latch 7475. |
| 003 | 000 | 000 | Codice dispositivo 000. |
| 004 | 166 | HLT | Alt |

Questo programma inserirà i dati dallo switch logico nell'accumulatore, poi metterà in uscita i dati dall'accumulatore su di un latch di uscita e finalmente si fermerà.

## SECONDO PROGRAMMA

Per inserire e mettere in uscita e in ingresso continuamente i dati acquisiti dal dispositivo d'ingresso 004, cambiate l'istruzione HLT con un'istruzione JMP che esegue un loop tornando a HI = 003 e LO = 000.

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci dati dal dispositivo d'ingresso 004. |
| 001 | 004 | 004 | Codice dispositivo 004. |
| 002 | 323 | OUT | Poni in uscita i dati sul dispositivo di uscita 000. |
| 003 | 000 | 000 | Codice dispositivo 000. |
| 004 | 303 | JMP | Salta in modo incondizionato alla locazione di memoria START. |
| 005 | 000 | START | Byte d'indirizzo LO di START. |
| 006 | 003 | – | Byte d'indirizzo HI di START. |

**TERZO PROGRAMMA**

Per memorizzare i dati in ingresso in una locazione di memoria ed aggiornare i contenuti di memoria ogni volta che viene inserito un nuovo byte dati a otto bit, userete il seguente programma:

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci dati dal dispositivo d'ingresso 004. |
| 001 | 004 | 004 | Codice dispositivo 004. |
| 002 | 323 | OUT | Poni in uscita i dati sul dispositivo in uscita 000. |
| 003 | 000 | 000 | Codice dispositivo 000. |
| 004 | 062 | STA | Memorizza i contenuti dell'accumulatore nella locazione di memoria data dai due byte seguenti. |
| 005 | 200 | STORE | Byte d'indirizzo LO di STORE. |
| 006 | 003 | – | Byte d'indirizzo HI di STORE. |
| 007 | 166 | HLT | Alt. |

Questo programma è simile al secondo programma, ma questa volta è stata aggiunta un'istruzione STA <B2> <B3> per permettervi di memorizzare i contenuti dell'accumulatore nella locazione di memoria STORE, che si trova a HI = 003 e LO = 200. Dopo che il programma si è arrestato, esaminate la locazione STORE per vedere se i dati d'ingresso dal dispositivo 004 sono presenti.

Cambiate l'impostazione dello switch, eseguite di nuovo il programma, ed esaminate ancora una volta la posizione di memoria STORE. Vi chiederete come possono essere memorizzati i dati quando sono stati già inviati verso il latch di uscita 000. A prima vista, pare che i dati d'ingresso siano stati "consumati" quando un byte dati viene trasferito da una posizione all'altra, esso viene "copiato" nella nuova posizione. I dati originali sono ancora presenti e non "consumati". Questa regola generale è valida per quasi tutti i trasferimenti di dati di un microcomputer, da registro a registro, dal registro alla memoria, dalla memoria al registro, dall'accumulatore al dispositivo di uscita, ecc.

## QUARTO PROGRAMMA

Questo programma è particolarmente interessante se avete un microcomputer MMD-1, nel quale la tastiera è la porta d'ingresso 000 (Vedere capitolo N.4).

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci i dati dalla tastiera sul microcomputer MMD-1. |
| 001 | 000 | 000 | Codice dispositivo 000. |
| 002 | 323 | OUT | Poni in uscita i dati sulla porta di uscita 000 sul micrcomputer MMD-1. |
| 003 | 000 | 000 | Codice dispositivo 000. |
| 004 | 303 | JMP | Salta in modo incondizionato alla locazione di memoria START. |
| 005 | 000 | START | Byte d'indirizzo LO di START. |
| 006 | 003 | – | Byte d'indirizzo HI di START. |

Quando eseguirete questo programma, sarete in grado di determinare la codificazione **di** ognuno dei quindici tasti sul microcomputer MMD-1. Il sedicesimo tasto è RESET, che entra a far parte dell'hardware direttamente sul chip 8224. La codificazione dei tasti può essere riassunta nel modo seguente:

| Tasto | Byte dei dati in ingresso D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Codice ottale |
|---|---|---|---|---|---|---|---|---|---|
| No key | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 160 (non signif.) |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 360 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 361 |
| 2 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 362 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 363 |
| 4 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 364 |
| 5 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 365 |
| 6 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 366 |
| 7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 367 |
| S | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 370 |
| C | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 372 |
| G | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 373 |
| H | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 374 |
| L | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 375 |
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 376 |
| B | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 377 |

Per quanto riguarda la suddetta tabella, dovreste osservare che: (a) i bit D4, D5 e D6 sono sempre a livello logico 1 dato che sono bit non collegati al bus dei dati; (b) quando viene premuto uno qualunque dei quindici tasti, il bit D7 va sempre a livello logico 1, indicando il bloccaggio del tasto e fungendo da bit di segnalazione; e (c) i bit D0, D1 e D2 corrispondono al codice ottale per il tasto della cifra ottale, ammesso che il bit D3 sia a livello logico 0.

## QUINTO PROGRAMMA

Questo programma somma uno ai contenuti dell'accumulatore, adatta i contenuti dell'accumulatore (DAA) secondo il sistema decimale, e poi mette in uscita il risultato decimale in codice binario, cioè due cifre BCD impaccate in un byte di dati a 8 bit, sulla porta di uscita 002.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 257 | XRA A | Azzera l'accumulatore |
| 001 | 306 | ADD, ADI | Somma il byte immediato all'accumulatore. |
| 002 | 001 | 001 | Byte immediato. |
| 003 | 047 | DAA | Adatta i contenuti risultanti nell'accumulatore secondo il sistema decimale. |
| 004 | 006 | MVI B | Sposta il byte di timing seguente nel registro B. |
| 005 | 040 | 040 | Byte di timing. |
| 006 | 315 | LOOP, CALL | Richiama il loop di delay di 10 ms DELAY posizionato in KEX. |
| 007 | 277 | DELAY | Byte d'indirizzo LO di DELAY. |
| 010 | 000 | – | Byte d'indirizzo HI di DELAY. |
| 011 | 005 | DCR B | Decrementa il registro B. |
| 012 | 302 | JNZ | Se il registro B *non* è uguale a 000, salta alla posizione di memoria LOOP altrimenti passa all'istruzione successiva. |
| 013 | 006 | LOOP | Byte d'indirizzo LO di LOOP. |
| 014 | 003 | – | Byte d'indirizzo HI di LOOP. |
| 015 | 323 | OUT | Poni in uscita le cifre BCD sulla porta di uscita 002. |
| 016 | 002 | 002 | Codice dispositivo 002. |
| 017 | 303 | JMP | Salta alla posizione di memoria ADD. |
| 020 | 001 | ADD | Byte d'indirizzo LO di ADD. |
| 021 | 003 | – | Byte d'indirizzo HI di ADD. |

Quando eseguite questo programma, osserverete i numeri BCD da 00 a 99 alla porta di uscita 002. Quando la porta raggiunge $99_{10}$ ritorna a 00 e ripete il processo di conteggio lento. Vorremmo farvi notare che non dovreste usare l'istruzione INR A per incrementare l'accumulatore immediatamente prima di un'istruzione DAA. L'istruzione ADI 001 porta allo stesso risultato e sistema i bit di riporto e di riporto ausiliario in modo che l'operazione DAA può essere eseguita nel modo corretto.

## INTRODUZIONE AGLI ESPERIMENTI

I seguenti semplici esperimenti illustrano le tecniche di I/O dell'accumulatore. Nel Capitolo 22, troverete esperimenti più ampi al riguardo.

| Esperimento N. | Commento |
|---|---|
| 1 | Un semplice circuito di ingresso-uscita di un microcomputer. Dimostrare l'uso dei latch 7475 e dei buffer a tre stati 8095 nell'I/O dell'accumulatore. |
| 2 | Ingresso-uscita del microcomputer sul microcomputer MMD-1. Dimostrare il funzionamento della tastiera sul microcomputer MMD-1. |
| 3 | Caratteristiche dell'istruzione DAA. Dimostrare che l'uso di un'istruzione DAA vi permette di addizionare numeri BCD impaccati a 8 bit. |

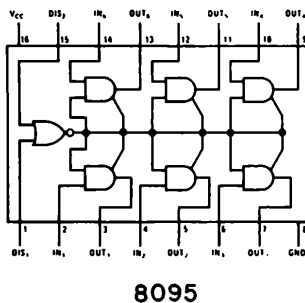
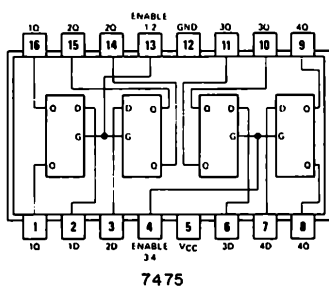
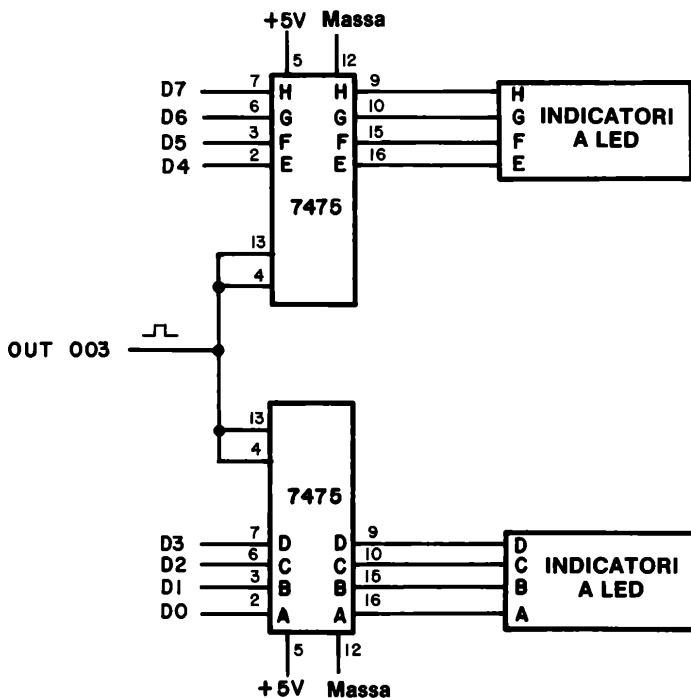
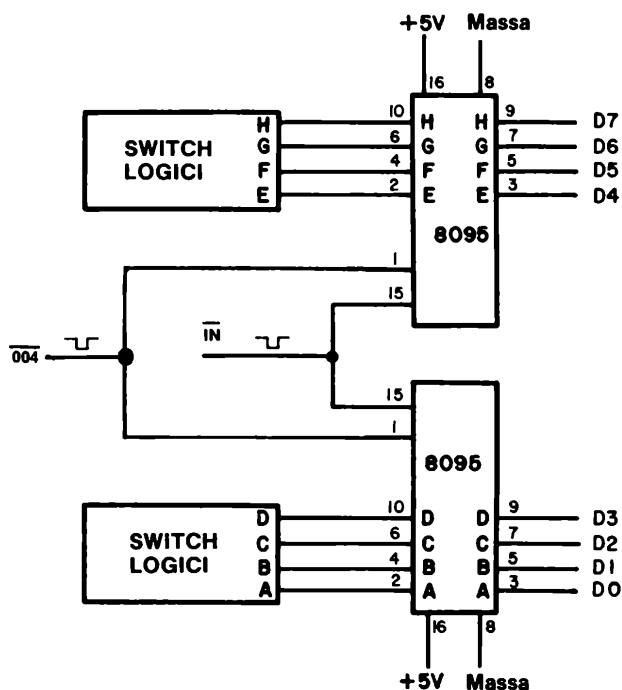
*Le porte di I/O dell'accumulatore che cablate nell'Esperimento N.1 verranno usate, con leggere modifiche, negli Esperimenti N. 1, e 3 del Capitolo N. 21. Non togliete dal vostro breadboard i circuiti delle porte di I/O 7475 e 8095.*

## ESPERIMENTO N. 1

## UN SEMPLICE CIRCUITO DI INGRESSO/USCITA DEL MICROCOMPUTER

**Scopo**

Lo scopo di questo esperimento è provare il comportamento di un semplice circuito di I/O del microcomputer basato sul buffer three-state 8095 e sul latch 7475.

**Configurazioni dei pin dei circuiti integrati**

7475

8095

**Schema del circuito**

**Programma**

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci i dati provenienti dallo switch logico alla porta d'ingresso 004. |
| 001 | 004 | 004 | Codice dispositivo 004. |
| 002 | 323 | OUT | Poni in uscita il contenuto dell'accumulatore sulla porta di uscita 003. |
| 003 | 003 | 003 | Codice dispositivo 003. |
| 004 | 303 | JMP | Salta in modo incondizionato alla locazione di memoria START. |
| 005 | 000 | START | Byte d'indirizzo LO di START. |
| 006 | 003 | – | Byte d'indirizzo HI di START. |

**Passo 1**
Questo esperimento vi serve come esperienza sia per cablare la porta di ingresso che quella di uscita. Se vi interessa cablare solo la porta d'ingresso, *non* cablate il circuito di latch 7475 e passate al passo 7.

Su un breadboard che ha spazio sufficiente per quattro circuiti integrati a 16 pin, cablate la porta di uscita 7475 e la porta di ingresso 8095. Ricordate che avrete bisogno anche dei circuiti di decodificazione per fornire gli impulsi di selezione dispositivo di ingresso e di uscita (vedere Capitolo 17). Se avete un solo chip decodificatore, potete passare al passo 7 e cablare solo i chip 8095.

**Passo 2**
Caricate il programma in memoria iniziando da HI = 003 e LO = 000.

**Passo 3**
Posizionate tutti gli switch logici sul livello logico 1. Eseguite il programma alla massima velocità. Che cosa osservate alla porta di uscita 003?

Tutti gli indicatori a LED della porta di uscita 003 sono accesi.

**Passo 4**
Mentre il microcomputer opera alla massima velocità, riportate ogni switch logico, uno alla volta, a livello logico 0. Nel farlo, spiegate che cosa osservate sugli indicatori a LED della porta di uscita.

Non appena uno switch logico è ritornato a livello logico 0, anche l'indicatore a LED corrispondente ritorna a livello logico 0. Vi è una corrispondenza tra uno switch logico ed un indicatore a LED.

**Passo 5**
Posizionate gli otto switch logici su HGFEDCBA o 11110101, o 365 in codice ottale. Eseguite il programma alla massima velocità e poi passate all'operazione passo a passo usando un circuito come quello descritto nell'Esperimento N.2 nel Capitolo 17. Cablate un circuito di bus monitor come quello descritto nell'Esperimento N.1 nel Capitolo 17. L'ingresso di abilitazione del latch dovrebbe essere a livello logico 0.

**Passo 6**
Proseguite con lo schema passo a passo e verificate la sequenza di byte che dovrebbe quindi apparire sul bus monitor. Notate che state eseguendo un loop continuo, perciò dovreste essere in grado di partire all'inizio tramite l'applicazione di molti impulsi passo a passo.

| Byte del bus di dati che appare sul monitor del bus | Commento |
|---|---|
| 333 | Ciclo macchina di FETCH per il codice dell'istruzione IN. |
| 004 | Ciclo macchina di FETCH per il byte <B2> dell'istruzione IN; (codice dispositivo della porta di ingresso). |
| 365 | Ciclo macchina di INGRESSO, durante il quale l'informazione presente sul bus dati bidirezionale esterno viene trasferita direttamente nell'accumulatore e il codice dispositivo appare sul bus d'indirizzo. Durante questo ciclo macchina viene generato anche un segnale di controllo $\overline{IN}$. (NOTA: L'impulso di selezione dispositivo d'ingresso 004 abilita la coppia di chip 8095 e fa sì che i dati dello switch logico appaiono sul bus dati. In questo caso, gli switch logici sono stati posizionati sul byte ottale, 365). |
| 323 | Ciclo macchina di FETCH per il codice dell'istruzione OUT. |
| 003 | Ciclo macchina di FETCH per il byte <B2> dell'istruzione OUT, (codice dispositivo della porta di uscita). |
| 365 | Ciclo macchina di USCITA, durante il quale i contenuti dell'accumulatore sono resi disponibili sul bus dati bidirezionale e il codice dispositivo appare sul bus dati. (NOTA: L'impulso di selezione dispositivo di uscita 003 abilita la coppia di chip 7475 e permette di effettuare il latch del byte ottale 365 che appare sul bus dati. È questo il byte di dati che era stato originariamente inserito durante l'istruzione IN). |
| 303 | Ciclo macchina di FETCH per il codice dell'istruzione JMP. |
| 000 | Ciclo macchina di FETCH per il byte <B2> dell'istruzione JMP. Questo è il byte d'indirizzo LO della locazione di memoria START. |
| 003 | Ciclo macchina di FETCH per il byte <B3> dell'istruzione JMP Questo è il byte d'indirizzo HI della locazione di memoria START. |

Continuando l'operazione passo a passo del microcomputer, sul bus dati verrà ripetuta la sequenza di byte suddetta. L'importante è il fatto che potete osservare il trasferimento dei dati fra l'accumulatore e un dispositivo di ingresso o di uscita. Quando lavorate con circuiti di interfaccia più complessi, potete avere un bus monitor e un circuito passo-passo per verificare che vengano trasferiti i dati giusti nel momento giusto.

**Passo 7**

Se non volete cablare un circuito latch 7475 e se avete un microcomputer MMD-1, potete trovarvi avvantaggiati dal fatto che vi sono tre porte di uscita 7475 sulla piastra. I codici dispositivo per queste porte sono 000, 001, e 002. Vi raccomandiamo di usare la porta di uscita 002 che richiede un cambiamento nel byte istruzioni da LO = 003 a LO = 002 Apportate questo cambiamento nel programma.

**Passo 8**
Posizionate gli otto switch logici sul livello logico 1. Eseguite il programma alla massima velocità. Che cosa osservate alla porta di uscita 002?

Tutti gli indicatori a LED sulla porta di uscita 002 sono accesi.

**Passo 9**
Mentre il microcomputer opera alla massima velocità, riportate tutti gli switch logici, uno per volta, al livello logico 0? Nel farlo, spiegate che cosa osservate sulla porta di uscita.

Non appena uno switch logico ritorna a livello 0 anche l'indicatore corrispondente ritorna a livello logico 0. Vi è una corrispondenza di ogni switch logico con un indicatore a LED.

**Passo 10**
Posizionate gli otto switch logici su HGFDCBA = 11110101, o 365 in codice ottale. Eseguite il programma alla massima velocità, poi passate all'operazione passo a passo usando un circuito come quello descritto nell'Esperimento N.2 nel Capitolo N. 17 Cablate un circuito bus monitor simile a quello descritto nell'Esperimento N.1 nel Capitolo N. 17. L'ingresso di abilitazione del latch dovrebbe essere a livello logico 0.

**Passo 11**
Avanzate passo a passo con l'esecuzione del programma e verificate la sequenza di byte data nel passo 7 di questo esperimento. *Ricordatevi che il ciclo macchina di FETCH per il codice dispositivo dell'istruzione di uscita pone il byte 002 sul bus di dati anzichè il byte 003. Perchè?*

Nel passo 7 avete cambiato il codice dispositivo della porta di uscita da 003 a 002. Perciò, il codice dispositivo 002 deve apparire sul bus dati durante il ciclo macchina di FETCH.

**Commento**
Questo esperimento completa molto di quello che avete imparato fino ad ora: la generazione e l'uso degli impulsi di selezione dispositivo, l'ingresso-uscita dell'accumulatore e la programmazione dell'8080A per le operazioni di ingresso-uscita. Sotto il controllo del software, sono state costruite ed usate una semplice porta di ingresso three-state ed una porta di uscita latch.

## ESPERIMENTO N. 2

## INGRESSO-USCITA DEL MICROCOMPUTER MMD-1

**Scopo**

Lo scopo di questo esperimento è di mostrare il funzionamento della tastiera del microcomputer MMD-1.

**Configurazione dei pin dei circuiti integrati**

Non è necessario nessun circuito di interfaccia, dato che sul microcomputer MMD-1 sono già presenti tutti i chip necessari - il latch 7475, il buffer three-state 8095 (SN74365) e il codificatore di priorità 74148. Le configurazioni dei pin dei chip 7475 e 8095 sono state date nell'esperimento precedente. Vi mostriamo ora la configurazione dei pin e la tabella della verità del chip 74148.

74148

SN54148, SN74148

FUNCTION TABLE

| INPUTS | | | | | | | | | OUTPUTS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EI | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | A2 | A1 | A0 | GS | EO |
| H | X | X | X | X | X | X | X | X | H | H | H | H | H |
| L | H | H | H | H | H | H | H | H | H | H | H | H | L |
| L | X | X | X | X | X | X | X | L | L | L | L | L | H |
| L | X | X | X | X | X | X | L | H | L | L | H | L | H |
| L | X | X | X | X | X | L | H | H | L | H | L | L | H |
| L | X | X | X | X | L | H | H | H | L | H | H | L | H |
| L | X | X | X | L | H | H | H | H | H | L | L | L | H |
| L | X | X | L | H | H | H | H | H | H | L | H | L | H |
| L | X | L | H | H | H | H | H | H | H | H | L | L | H |
| L | L | H | H | H | H | H | H | H | H | H | H | L | H |

**Schema del circuito**

Lo schema della sezione di ingresso-uscita del microcomputer MMD-1 è presentato nella pagina seguente per gentile concessione della Gernsback Publications, Inc. (Tutti i diritti sono riservati).

**Programma**

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci i dati dalla tastiera del microcomputer MMD-1. |
| 001 | 000 | 000 | Codice dispositivo 000. |
| 002 | 323 | OUT | Poni in uscita i dati sulla porta di uscita 002 sul microcomputer MMD-1. |
| 003 | 002 | 002 | Codice dispositivo 002. |
| 004 | 303 | JMP | Salta in modo incondizionato alla posizione di memoria START. |

| | | | |
|---|---|---|---|
| 005 | 000 | START | Byte d'indirizzo LO di START. |
| 006 | 003 | — | Byte d'indirizzo HI di START. |

**Passo 1**
Studiate il seguente schema della sezione di ingresso-uscita del microcomputer MMD-1.

Da notare quanto segue:

- La sezione di ingresso-uscita è un esempio dell'I/O dell'accumulatore. Le indicazioni che abbiamo seguito per arrivare a questa conclusione sono state i segnali di controllo $\overline{IN}$ e $\overline{OUT}$ sul lato sinistro dello schema.

- Il circuito decodificatore di I/O consiste di un chip decodificatore 74L42 e di gate NOR a 2 ingressi 7402. Per ulteriori dettagli, osservate la figura 17-7 e consultate il testo ad essa relativo, nonchè l'Esperimento N. 5 del Capitolo N. 17.

- Vi sono tre porte di uscita a 8 bit:

  - Porta di uscita 000, che consiste nei latch 7475 IC26 e IC27

  - Porta di uscita 001, che consiste nei latch 7475 IC24 e IC25

  - Porta di uscita 002, che consiste nei latch 7475 IC28 e IC29

- C'è una porta di ingresso a 5 bit, che viene collegata in ultimo alla tastiera:

  - Porta di ingresso 000, che consiste in un chip 8095, IC31

- La coppia di chip decodificatori di priorità 74148 e il gate NAND a 2 ingressi 7400 fanno sì che i circuiti dedecodifichino quindici tasti della tastiera. Quando è premuto un solo tasto, l'ingresso al 74148 corrispondente a quel tasto va a livello logico 0 e l'uscita GS sul chip 74148 va a livello logico 0. Il codice ottale a tre bit del tasto premuto appare ai pin di uscita A, B e C sul chip 74148. Ricordate che il sedicesimo tasto, RESET, è una funzione hardware e non genera nessun codice.

**Passo 2**
Caricate il programma in memoria iniziando da HI = 003 e LO = 000.

**Passo 3**
Eseguite il programma alla massima velocità. Quando non premete nessun tasto della tastiera, quali bit si accendono sulla porta di uscita 002?

I bit D4, D5 e D6 sono a livello logico 1; gli altri cinque bit sono a livello logico 0.

**Passo 4**
Premete qualunque tasto della tastiera eccetto il tasto RESET. È vero che il bit D7 sulla porta di uscita 002 va sempre a livello logico 1 quando viene premuto uno qualunque degli altri quindici tasti?

Sì. Perché questo è il bit che indica al programma del monitor KEX che è stato premuto un tasto.

Una volta che il microcomputer apprende che è stato premuto un tasto, procede poi a determinare di quale tasto si tratta. Nella routine che svolge questo compito c'è un ritardo di tempo di 10 ms per eliminare il problema del rimbalzo del contratto. Il programma di ingresso da KEX della tastiera è elencato alla fine di questo esperimento.

**Passo 5**
Fin qui, avete osservato che i bit D4, D5 e D6 rimangono a livello logico 1 qualunque tasto venga premuto e che il bit D7 va a livello logico 1 in qualunque momento venga premuto un tasto. Dovete ora determinare la funzione degli altri quattro bit, da D0 a D3.

Premete uno dopo l'altro i tasti da 0 a 7 ed osservate la configurazione di bit alla porta di uscita 002 da D0 a D2. Che cosa vedete?

La configurazione di bit da D0 a D2 corrisponde al codice a tre bit dei tasti da 0 a 7. Così, il tasto 5 avrà il codice a tre bit, 101.

**Passo 6**
Premete i tasti H, L, G, S, A, B e C e spiegate che cosa osservate al bit D3 sul latch di uscita. Spiegate anche il significato dei bit da D0 a D2 quando questi tasti sono premuti.

Quando uno dei tasti H, L, G, S, A, B, o C è premuto, il bit D3 va sempre a livello logico 1. Gli altri tre bit, da D0 a D2, decodificano il tasto che è premuto. Avete ora verificato la codificazione della tastiera. Tale codificazione può essere riassunta dalla seguente tabella della verità, che dà gli stati logici dei bit quando il tasto è premuto.

| Tasti | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 6 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| S | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| C | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| G | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| H | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| L | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| B | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Passo 7**
Quando i tasti della tastiera sono attivati, il codice viene inserito e poi messo in uscita dall'8080A sui LED. È importante ricordare che questo processo di trasferimento dati è sotto controllo del software.

Cambiate il byte del codice dispositivo a LO = 001 in 005 nel programma, poi eseguitelo a 750 kHz. Il programma opera come faceva in precedenza? Perchè?

No. L'indirizzo dispositivo per l'istruzione di ingresso è stato cambiato in modo che la tastiera non viene più selezionata.

**Listing della subroutine KBRD**

L'ingresso istantaneo dei dati della tastiera può non coincidere con i codici mostrati precedentemente a causa dei rimbalzi di contatto negli interruttori meccanici. Un programma "filtro" antirimbalzo è disponibile nella memoria di sola lettura programmabile KEX (PROM) a partire dall'indirizzo 000 315. Questo programma, chiamato KBRD, può essere chiamato con un richiamo della subroutine, e ritorna con il valore del codice tasti situato nell'accumulatore. Se volete usare questa subroutine assicuratevi di avere riservato un'area per lo stack. Il listing della subroutine di ingresso della tastiera è il seguente:

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 315 | 333 | KBRD, IN | /Input dai keyboard encoder |
| 000 316 | 000 | 000 | |
| 000 317 | 267 | ORA A | /Set dei flag |
| 000 320 | 372 | JM | /Salta all'indietro se l'ultimo tasto |
| 000 321 | 315 | KBRD | non è stato rilasciato |
| 000 322 | 000 | Ø | |
| 000 323 | 315 | CALL | /Attesa di 10 ms |
| 000 324 | 277 | TIMOUT | |
| 000 325 | 000 | Ø | |
| 000 326 | 333 | FLAGCK, IN | |
| 000 327 | 000 | 000 | |
| 000 330 | 267 | ORA A | |
| 000 331 | 362 | JP | /Salta all'indietro in attesa che un nuovo |
| 000 332 | 326 | FLAGCK | tasto venga premuto |
| 000 333 | 000 | Ø | |
| 000 334 | 315 | CALL | /Attesa di 10 ms |
| 000 335 | 277 | TIMOUT | |
| 000 336 | 000 | Ø | |

```
000 337        333          IN
000 340        000          000
000 341        267          ORA A
000 342        362          JP          /Salta all'indietro se il nuovo tasto non è
000 343        326          FLAGCK      /ancora premuto (falso allarme)
000 344        000          0
000 345        346          ANI         /Maschera tutti i bit tranne il codice tasti
000 346        017          017
000 347        345          PUSH H      /Salvataggio dei registri H e L
000 350        046          MVI H       /Azzera il registro H
000 351        000          000
000 352        306          ADI         /Somma l'indirizzo dell'inizio della
                                         tabella
000 353        360          360         /al codice tasti
000 354        157          MOV L,A
000 355        176          MOV A,M     /Preleva un nuovo valore dalla tabella
000 356        341          POP H       /Ripristina di nuovo i registri H e L
000 357        311          RET
```

La seguente tabella di traduzione converte il codice generato dai tasti nel codice usato dal programma KEX

```
000 360        000     TABLE, 000
000 361        001            001
000 362        002            002
000 363        003            003
000 364        004            004
000 365        005            005
000 366        006            006
000 367        007            007
000 370        013            013      /S
000 371        000            000      /Questo codice non può essere
                                        generato
000 372        017            017      /C
000 373        012            012      /G
000 374        010            010      /H
000 375        011            011      /L
000 376        015            015      /A
000 377        016            016      /B
```

**Listing della subroutine timout**

| Indirizzo di di memoria | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 277 | 365 | TIMOUT,PUSH PSW | /Salvataggio dell'accumulatore e dei /flag |
| 000 300 | 325 | PUSH D | /Salvataggio della coppia di registri D |

| | | | |
|---|---|---|---|
| 000 300 | 325 | PUSH D | /Salvataggio della coppia di registri D |
| 000 301 | 021 | LXI | /Carica D ed E con valore da decrementare |
| 000 302 | 046 | 046 | |
| 000 303 | 001 | 001 | |
| 000 304 | 033 | MORE, DCX D | /Decrementa la coppia di registri D |
| 000 305 | 172 | MOV A, D | /Sposta D verso A |
| 000 306 | 263 | ORA E | /OR di E con A |
| 000 307 | 302 | JNZ | /Il registro A è = 000? Se no, salta |
| 000 310 | 304 | MORE | /a MORE a LO = 304 e HI = 000. |
| 000 311 | 000 | Ø | /Altrimenti, passa all'istruzione seguente |
| 000 312 | 321 | POS D | /Ripristino della coppia di registri D |
| 000 313 | 361 | POP PSW | /Ripristino dell'accumulatore e dei flag |
| 000 314 | 311 | RET | /Rientra dalla subroutine TIMOUT |

I programmi suddetti sono stati scritti nel modo in cui verrebbero presentati dall'8080A resident assembler/editor della Tychon Inc.

## ESPERIMENTO N. 3

## CARATTERISTICHE DELL'ISTRUZIONE DAA

**Scopo**

Lo scopo di questo esperimento è esaminare alcune delle caratteristiche dell'istruzione DAA.

**Programma**

| Indirizzo di memoria LO | Byte di istruzioni | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 000 | BEGIN, NOP | Nessuna operazione |
| 001 | 257 | XRA A | Azzera l'accumulatore |
| 002 | 306 | ADD, ADI | Somma il byte seguente all'accumulatore |
| 003 | 001 | 001 | Byte di dati |
| 004 | 047 | DAA | Aggiustamento decimale dei contenuti dell'accumulatore |
| 005 | 006 | MVI B | Poni il byte di timing seguente nel registro B |
| 006 | 040 | 040 | Byte di timing |
| 007 | 315 | REPEAT, CALL | Richiama la routine DELAY di ritardo di 10 ms, posizionata in KEX EPROM |
| 010 | 277 | DELAY | Byte d'indirizzo LO di DELAY |
| 011 | 000 | – | Byte d'indirizzo HI di DELAY |
| 012 | 005 | DCR B | Decrementa il registro B |
| 013 | 302 | JNZ | Se il registro B *non* è uguale a 000, salta alla posizione di memoria REPEAT: altrimenti, passa all'istruzione successiva |
| 014 | 007 | REPEAT | Byte d'indirizzo LO di REPEAT |
| 015 | 003 | – | Byte d'indirizzo HI di REPEAT |
| 016 | 323 | OUT | Poni in uscita la coppia di digit BCD impaccati, sulla porta di uscita 002 |
| 017 | 002 | 002 | Codice dispositivo 002 |
| 020 | 303 | JMP | Salta alla posizione di memoria ADD |
| 021 | 002 | ADD | Byte d'indirizzo LO di ADD |
| 022 | 003 | – | Byte d'indirizzo HI di ADD |

**Passo 1**
Questo esperimento usa la porta di uscita 002 sul microcomputer MMD-1. Il nostro obiettivo è dimostrare come viene usata l'istruzione DAA per facilitare le operazioni in aritmetica BCD.

Che cosa s'intende per una coppia di "digit BCD impaccati"?

**Passo 2**
Caricate il programma nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 000. Eseguite il programma. Osservate che cosa accade immediatamente dopo che il conteggio BCD nella porta di uscita 002 raggiunge $99_{10}$. Riassumete di seguito le vostre osservazioni.

Abbiamo osservato che i conteggi che seguivano immediatamente 99 erano 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10 . . . . In altre parole, la porta di uscita "si rovescia" e ricomincia a conteggiare da 00. Questo comportamento è quello che ci aspettavamo.

**Passo 3**
Ora cambiate i seguenti byte istruzioni nel programma:

| | | | |
|---|---|---|---|
| 002 | 000 | NOP | Nessuna operazione |
| 003 | 074 | INR A | Incrementa l'accumulatore di uno |
| 006 | 200 | 200 | Byte di timing |

Eseguite il programma, con le modifiche suddette, e spiegate che cosa osservate sulla porta di uscita 002 che segue immediatamente $99_{10}$

Abbiamo osservato una strana sequenza di cifre che vengono meglio elencate in codice esadecimale: 00, 61, C2, 23, 84, E5, 46, A, 08, 69, D0, 31, 92, ecc. La cifra BCD meno significativa era esatta, ma la cifra BCD più significativa variava in un modo imprevedibile.

**Passo 4**

Perché avete avuto problemi quando l'istruzione INR A ha preso il posto dell'istruzione ADI 001? Dopo tutto, entrambe encrementano i contenuti dell'accumulatore di uno.

L'istruzione ADI 001 coinvolge sia il flag di di riporto che quello di riporto ausiliario, a seconda del risultato dell'operazione. Al contrario, *l'istruzione INR A coinvolge solo il flag di riporto ausiliario.* Affinchè l'istruziione DAA operi nel modo corretto, sia il riporto che il riporto ausiliario devono rispondere correttamente ad un'istruzione aritmetica che precede immediatamente l'istruzione DAA. Questo non è il caso dell'istruzione INR A.

**Passo 5**

Considerate i seguenti passi iniziali nel programma:

| | | | |
|---|---|---|---|
| 000 | 076 | MVI A | Poni il byte seguente nell'accumulatore |
| 001 | AUG | AUG | Byte di dati AUG che funge da augendo per l'addizione |
| 002 | 306 | ADI | Somma il byte seguente all'accumulatore |
| 003 | ADD | ADD | Byte di dati ADD che funge da addendo per l'addizione |

Una volta che questa addizione è stata eseguita, ed anche la routine di delay, e la somma messa in uscita sulla porta 002, il microcomputer viene arrestato attraverso l'uso del seguente byte di istruzioni,

| | | | |
|---|---|---|---|
| 020 | 166 | HLT | Alt |

I termini, augendo e addendo, sono stati definiti da Graf nel modo seguente:

*Addendo* — Una quantità che, se aggiunta ad un'altra quantità (chiamata l'augendo), produce un risultato chiamato somma.

*Augendo* — In un'addizione aritmetica, il numero che aumenta aggiungendovi un altro numero (chiamato addendo).

Nella suddetta sezione di programma di questa fase, i byte di dati AUG e ADD devono essere numeri binari o numeri BCD impaccati?

La risposta a questa domanda è determinante per l'operazione dell'istruzione DAA, quindi pensateci attentamente. Per favore scrivete qui di seguito la risposta.

La risposta corretta è la seguente:

- Se i quattro passi *non* sono seguiti da un'istruzione DAA, i byte di dati AUG e ADD vengono trattati come numeri binari a 8 bit e viene eseguita un'addizione binaria diretta per produrre un numero binario come somma.

- Se i quattro passi sono seguiti da un'istruzione DAA, *i byte di dati AUG e ADD sono trattati come numeri BCD impaccati a 8 bit e viene eseguita automaticamente una correzione BCD all'interno dell'8080A per produrre un numero BCD impaccato come somma.*

Questa è una distinzione molto importante e che va ricordata. Con il microprocessore 8080A, potete eseguire semplici operazioni aritmetiche BCD, il che significa che augendo, addendo e somma sono considerati tutti numeri BCD impaccati. Dovreste comunque ricordare che l'8080A e quasi tutti gli altri microcomputer sono processori binari. Tutti i dati vengono trattati come 1 e 0 binari. Per eseguire operazioni aritmetiche BCD occorre molta attenzione nel programmare e nell'usare l'istruzione DAA. Non deve essere considerato come un convertitore da binario a decimale!

**Passo 6**
Cambiate i byte di istruzione agli indirizzi di memoria LO 000, 002, e 020 con quelli indicati nel passo 5. Eseguite le seguenti addizioni fra i byte di dati BCD AUG e ADD, e confrontate la SOMMA che osservate sulla porta di uscita 002 con quella data nella tabella seguente.

| AUG (BCD impaccato) | ADD (BCD impaccato) | SUM (BCD impaccato) | Porta di uscita 002 (BCD impaccato) |
|---|---|---|---|
| 10 | 00 | 10 | |
| 10 | 10 | 20 | |
| 10 | 15 | 25 | |
| 15 | 19 | 34 | |
| 27 | 28 | 55 | |
| 33 | 48 | 81 | |
| 38 | 75 | 13* | |
| 99 | 99 | 98* | |

* Riporto = 1 per queste somme

Che cosa concludete?

Dovreste concludere che quando un'operazione di addizione e immediatamente seguita da un'istruzione DAA, i byte di dati addizionati devono essere considerati quantità BCD impaccate e la SOMMA è una quantità BCD impaccata con o senza riporto. Addizioni BCD vengono eseguite dal microcomputer attraverso un'addizione binaria ed una correzione BCD chiamata accumulator DAA. Il chip 8080A intel/national può eseguire solo addizioni BCD. Le sottrazioni richiedono accorgimenti particolari, dato che il riporto ausiliario non è coinvolto dalle istruzioni SUB e SBB. Il chip 8080A ha un bit di flag in più, SUB, che vi permette di eseguire sottrazioni BCD impaccate usando SUB, SBB, e SBI.

**Passo 7**
Eliminate l'istruzione DAA sostiuendo il seguente byte di istruzione NOP:

| | | | |
|---|---|---|---|
| 004 | 000 | NOP | Nessuna operazione |

Eseguite le seguenti addizioni fra AUG e ADD e confrontate la SOMMA che osservate sulla porta di uscita 002 con quella data nella tabella seguente.

| AUG (binario) | ADD (binario) | SOMMA (binario) | Porta di uscita 002 (binario) |
|---|---|---|---|
| 10 | 00 | 10 | |
| 10 | 10 | 20 | |
| 10 | 15 | 25 | |
| 15 | 19 | 2E | |
| 27 | 28 | 4F | |
| 33 | 48 | 7B | |
| 38 | 75 | AD | |
| 99 | 99 | 32* | |

* Riporto: 1 per questa somma

Notate che gli ingressi delle colonne AUG e ADD sono dati in codice esadecimale e non in codice ottale o decimale. Uno degli obiettivi di questo esperimento è darvi un po' di pratica nel convertire quantità decimali ed esadecimali in codice ottale, e viceversa.

Che cosa concludete dalla suddetta tabella?

Dovreste concludere che quando un'addizione di due numeri non è immediatamente seguita da un'istruzione DAA, i byte di dati AUG e ADD nonchè la SOMMA devono essere considerati regolari numeri binari a 8 bit. Il microcomputer esegue ora operazioni aritmetiche binarie, ma la somma non è stata corretta per dare una risposta BCD.

Questo esperimento dimostra che potete inserire e mettere in uscita dati BCD sia binari che impaccati ed eseguire addizioni sia binarie che BCD usando tali dati. Abbiamo simulato l'ingresso dei dati BCD usando l'istruzione MVI A.

## DOMANDE RIEPILOGATIVE

Le seguenti domande vi aiuteranno a rivedere le tecniche di ingresso/uscita dell'accumulatore.

1. Che cosa si intende dire con il termine "ingresso/uscita tramite l'accumulatore"?

2. Che differenza esiste fra i chip usati per le uscite dei microcomputer e quelli usati per gli ingressi? Spiegate perchè i chip che servono a due usi differenti devono essere diversi in funzione.

3. Perchè sono importanti nei circuiti di uscita dei microcomputer la capacità di pilotaggio in uscita di un microprocessore e le caratteristiche del fan-in dei circuiti di intefaccia?

4. In base alle informazioni date nel quinto programma e nell'esperimento N. 3, spiegate le caratteristiche dell'istruzione DAA nell'addizione di una coppia di numeri a 8 bit.

**RISPOSTE**

1. Ingresso/uscita tramite l'accumulatore è un termine collegato con i sistemi a microcomputer 8080. Le istruzioni I/O sono IN e OUT e il trasferimento dei dati avviene fra il dispositivo di I/O e l'accumulatore all'interno del chip 8080.

2. Nell'uscita dei microcomputer, l'obiettivo è "catturare" i dati che sono messi in quel momento in uscita per un breve intervallo di tempo. Ciò avviene usando un latch a 8 bit che viene abilitato durante quel breve intervallo di tempo.

3. Nell'ingresso dei microcomputer, l'obiettivo è inserire dati TTL stabili nel chip durante l'intervallo di tempo in cui il dispositivo di uscita esterno ha il controllo sul bus dati. In tutti gli altri momenti, il dispositivo di uscita non dovrebbe influenzare il bus di dati. Tali obiettivi vengono raggiunti usando un buffer three-state a 8 bit.

4. In assenza di un'istruzione DAA, quando due numeri a 8 bit vengono addizionati usando un'istruzione ADD, ADC, ADI o simili, la somma di due numeri è in codice binario e l'augendo e l'addendo vengono considerati quantità binarie. Quando un'istruzione DAA segue immediatamnete un'istruzione ADD, ADC o ADI, l'augendo, l'addendo e la somma devono essere considerati tutti numeri BCD impaccati a 2 cifre. In altre parole, potete eseguire addizioni binarie pure o addizioni BCD pure a seconda se l'istruzione DAA è assente o presente.

# CAPITOLO 21

# INTRODUZIONE ALLE TECNICHE DI INGRESSO/USCITA MEMORY MAPPED

## INTRODUZIONE

Nell'I/O memory mapped, un dispositivo di ingresso/uscita viene trattato come una locazione di memoria e si usano le istruzioni di trasferimento in memoria quali MOV, STAX, LDAX, STA, LDA, SHLD e LHLD per inserire ed ottenere i dati. Qualunque registro universale può essere la sorgente o la destinazione dei dati di I/O in mappa di memoria. In questo capitolo, userete semplici programmi di I/O memory mapped e cablerete semplici circuiti di interfaccia di ingresso/uscita che vi permetteranno di trasferire i dati con l'aiuto delle istruzioni che si riferiscono alla memoria.

## OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Riassumere le differenze fra le tecniche di I/O relative all'accumulatore e di I/O memory mapped.
- Disegnare un circuito che può essere usato per generare impulsi di selezione di indirizzi di memoria.
- Spiegare come vengono usati gli impulsi di selezione indirizzi di memoria per raggiungere l'obiettivo dell'ingresso/uscita memory mapped.
- Cablare un semplice circuito di ingresso memory mapped.
- Cablare un semplice circuito di uscita memory mapped.
- Confrontare le caratteristiche di ingresso memory mapped delle tre istruzioni seguenti: LDA, LDAX e MOV A,M.
- Confrontare le caratteristiche di uscita memory mapped delle tre istruzioni seguenti: STA, STAX e MOV M,A.

## I/O MEMORY MAPPED E I/O TRAMITE ACCUMULATORE

Un dispositivo di ingresso/uscita può essere una telescrivente, un display, un tubo a raggi catodici (CRT), uno strumento di laboratorio, un microcomputer, o un piccolo dispositivo digitale come un circuito integrato. Tutti i dispositivi di I/O possono scambiare i dati fra il microprocessore 8080A per mezzo dello *I/O tramite accumulatore o dell'I/O memory mapped,* tecniche simili fra di loro come concetti di base. Confrontiamo queste due tecniche di I/O nelle tabelle 9 e 10.

*Tabella 9.* Riassunto delle caratteristiche dell'I/O tramite accumulatore.

| | |
|---|---|
| Istruzioni 8080A: | OUT <B2><br>IN <B2> |
| Segnali di controllo: | $\overline{\text{OUT}}$<br>$\overline{\text{IN}}$ |
| Trasferimento dei dati: | Fra l'accumulatore e il dispositivo di I/O. |
| Decodificazione dispositivi: | Un codice dispositivo a 8 bit, da A0 ad A7 o da A8 ad A15, che è il byte <B2> nell'istruzione IN o OUT. Raccomandiamo che sia decodificato in assoluto, cioè che tutti gli otto bit vengano usati per designare, o decodificare, uno specifico dispositivo di I/O. |
| Terminologia: | I processi di I/O verranno chiamati *ingresso e uscita*. Il segnale decodificato che fornisce segnali di abilitazione al dispositivo di I/O verrà chiamato *impulso di selezione dispositivo.* |

*Tabella 10.* Riassunto delle caratteristiche dell'I/O memory mapped.

| | | | |
|---|---|---|---|
| Istruzioni 8080A: | MOV B,M | MOV M,H | ANA M |
| | MOV C,M | MOV M,L | XRA M |
| | MOV D,M | MOV M,A | ORA M |
| | MOV E,M | STAX B | CMP M |
| | MOV H,M | STAX D | INR M |
| | MOV L,M | LDAX B | DCR M |
| | MOV A,M | LDAX D | MVI M |
| | MOV M,B | ADD M | STA <B2> <B3> |
| | MOV M,C | ADC M | LDA <B2> <B3> |
| | MOV M,D | SUB M | SHLD<B2><B3> |
| | MOV M,E | SBB M | LHLD<B2><B3> |
| Segnali di controllo: | $\overline{\text{MEMR}}$<br>$\overline{\text{MEMW}}$ | | |
| Trasferimento dei dati: | Fra il dispositvo di I/O posto nella mappa di memoria e i registri B,C,D,E,H,L o l'accumulatore (registro A). | | |
| Decodificazione dei dispositivi: | Un codice dispositivo a 16 bit, da A0 a A15, che è contenuto o nella coppia di registri H, o nella coppia di registri B, o nella coppia di registri D, o nei byte <B2> e <B3> per le istruzioni STA, LDA, SHLD, o LHLD. | | |

In alcuni casi, è utile e conveniente riservare l'area di memoria superiore a 32K per gli indirizzi di I/O posti in mappa di memoria; quando A15 sul bus di indirizzi è al livello logico 1, esiste l'ingresso/uscita memory mapped. I bit da A0 ad A7 possono essere usati per decodificare uno specifico dispositivo di I/O quando A15=1. Il dispositivo di I/O è fatto in modo da sembrare un'unica locazione di memoria a 8 bit e le istruzioni che si riferiscono alla memoria vengono naturalmente usate per leggere dallo o scrivere nello specifico dispositivo di I/O memory mapped.

Terminologia: I processi di I/O memory mapped saranno chiamati *lettura e scrittura* anzichè ingresso e uscita. Il segnale decodificato che fornisce segnali di abilitazione ad un dispositivo di I/O collocato in mappa di memoria sarà chiamato *impulso di selezione indirizzo* anzichè impulso di selezione dispositivo.

Si possono chiaramente vedere i vantaggi delle tecniche di I/O memory mapped confrontando le Tabelle 9 e 10. Il trasferimento dei dati può avvenire fra il dispositivo di I/O e uno qualunque dei registri universali all'interno del chip 8080A. L'indirizzo di memoria a 16 bit è stato precedentemente memorizzato nella coppia di registri H, il trasferimento dei dati può avvenire velocemente se si usa un'istruzione MOV r, M o MOV M, r. In principio, possono venire indirizzati molti più dispositivi con le tecniche di I/O memory mapped che con le tecniche di I/O tramite accumulatore. Alla fine, i trasferimenti di dati a due byte in una sola istruzione sono possibili usando le istruzioni SHLD e LHLD.

È importante ricordare che le tecniche di I/O memry mapped e di I/O tramite l'accumulatore fondamentalmente non sono diverse l'una dall'altra. In ogni caso, un segnale di controllo indica se si tratta di un'operazione di ingresso o di uscita. Inoltre, il bus d'indirizzo deve essere sempre decodificato per identificare uno specifico dispositivo di I/O. Infine, in ogni caso il trasferimento dei dati avviene in un ciclo macchina durante l'esecuzione dell'istruzione. Per l'I/O tramite l'accumulatore, questo ciclo macchina genera, con l'aiuto di un latch di stato, i segnali di controllo $\overline{IN}$ e $\overline{OUT}$; per l'I/O memory mapped, vengono invece generati i segnali $\overline{MEMR}$ e $\overline{MEMW}$. Potete osservare il trasferimento dei dati sul bus di dati bidirezionale con l'aiuto di un bus monitor. (vedi Esperimento n. 1 del Capitolo N. 17).

Lavorando con le tecniche di I/O memory mapped, abbiamo osservato che sono facili da usare. La maggior parte dei problemi possono essere attribuiti alla mancanza di decodificazione in assoluto di tutto il bus d'indirizzi a 16 bit. Quando si indirizza un dispositivo di I/O localizzato in mappa di memoria, non è sufficiente decodificare i bit da A0 ad A9, dato che questi stessi bit vengono usati in qualunque sistema microcomputer basato sull'8080A che abbia almeno 1K di memoria indirizzabile. Perciò, se volete usare le tecniche di I/O memory mapped dovreste decidere di decodificare alcuni dei bit più alti sul bus d'indirizzi, specialmente i bit da A13 ad A15.

## COME SI GENERANO IMPULSI DI SELEZIONE INDIRIZZI LOCALIZZATI IN MAPPA DI MEMORIA

Per generare un impulso di selezione indirizzi occorrono due tipi di informazioni dal microcomputer 8080A:

1. Un codice di identificazione a più bit, chiamato *indirizzo di memoria,* del dispositivo esterno di I/O.

2. Un impulso di sincronizzazione ad un solo bit, o $\overline{MEMR}$ o $\overline{MEMW}$, che sincronizza la decodificazione del codice dispositivo.

L'origine di entrambi i tipi di informazione è nel software, cioè nelle istruzioni con riferimento alla memoria, quali MOV r,M, STAX B, MOV M,r, STAX D, ADD M, MVI M, LDAX D, CMP M, ecc. Tali istruzioni fanno sì che il microprocessore 8080A metta un indirizzo a 16 bit sul bus d'indirizzo e generi anche o un segnale di controllo di lettura in memoria, $\overline{MEMR}$ (per i dati di memoria che sono inseriti nel chip 8080A), o un segnale di controllo di scrittura in memoria, $\overline{MEMW}$ (per i dati di memoria che sono messi in uscita dal chip 8080A).

In altre parole, come per l'I/O tramite l'accumulatore, durante la generazione di un impulso di selezione indirizzi di I/O memory mapped, sia il bus d'indirizzo che il bus di controllo sono attivi. Spetta a voi decodificare in modo corretto i segnali su questi due bus per produrre impulsi di selezione indirizzi unici che possono essere usati per trasferire i dati fra il dispositivo esterno di I/O e i registri interni dell'8080A.

La Figura 21-1 mostra una tecnica di decodifica comunemente usata per gli impulsi di selezione indirizzi in mappa di memoria. Notate la rassomiglianza fra questa figura e la Figura 17-2 del Capitolo N. 17. Il decodificatore 74154 è abilitato da due segnali: il complemento del bit del bus d'indirizzo A-15, e $\overline{MEMR}$ o $\overline{MEMW}$. Le Tabelle della verità del processo di abilitazione del chip sono date nella Tabella 11. Osservate che il decodificatore 74154 è abilitato solo quando il bit del bus d'indirizzo A-15 è a livello logico 1. I dati sono messi in ingresso solo quando $\overline{MEMR}$ è a livello logico 0 e sono messi in uscita solo quando $\overline{MEMW}$ è a livello logico 0. Nel momento in cui non viene trasferito nessun dato di memoria, sia $\overline{MEMR}$ che $\overline{MEMW}$ sono a livello logico 1.

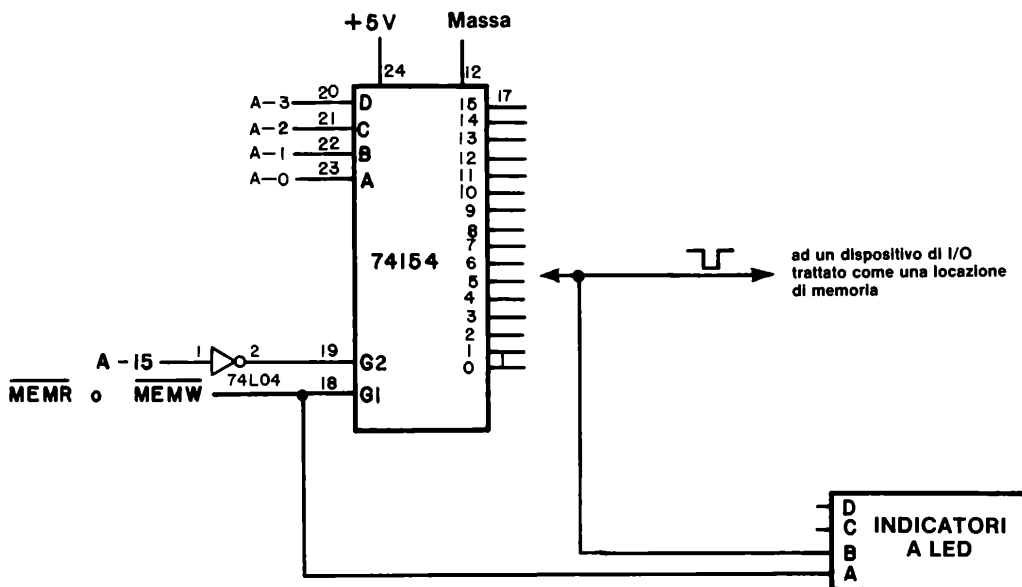



*Figura 21-1.* Circuito di decodifica per la generazione di sedici diversi impulsi d'indirizzo in mappa di memoria. Il byte d'indirizzo HI è 200 e il byte d'indirizzo LO va da 000 a 017, in codice ottale. Questo non è un circuito di decodifica in assoluto per il bus d'indirizzo a 16 bit.

*Tabella 11.* Tabelle della verità del circuito di decodifica mostrato in Figura 21-1.

| A-15 | $\overline{MEMR}$ | Comportamento del decodificatore | A-15 | $\overline{MEMW}$ | Comportamento del decodificatore |
|---|---|---|---|---|---|
| 0 | 0 | disabilitato | 0 | 0 | disabilitato |
| 0 | 1 | disabilitato | 0 | 1 | disabilitato |
| 1 | 0 | *generazione impulso di lettura* | 1 | 0 | *generazione impulso di scrittura* |
| 1 | 1 | disabilitato | 1 | 1 | disabilitato |

Nella Figura 21-2, dieci dei sedici bit del bus d'indirizzo vengono decodificati da una coppia di decodificatori 74154. Per il circuito mostrato, vengono prodotti sedici impulsi di selezione indirizzi di I/O in mappa di memoria, iniziando da HI = 300 e LO = 000 e finendo con HI = 300 e LO = 017. Il decodificatore 74154 N. 2 è abilitato solo quando sia A-14 che A-15 sono a livello logico 1. Uno qualunque dei sedici pin di uscita sul decodificatore N. 2 può essere usato per abilitare l'ingresso G2 del decodificatore N. 1.

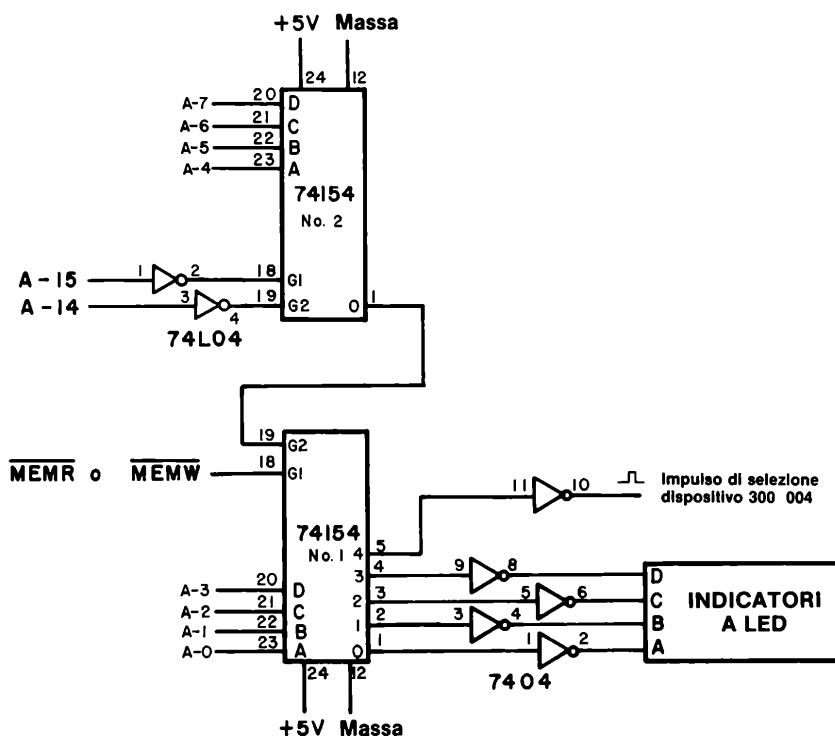



*Figura 21-2.* Circuito di decodifica che genera sedici diversi impulsi di selezione indirizzi di I/O in mappa di memoria iniziando dall'indirizzo di memoria HI = 300 e LO = 000. Dieci dei sedici bit del bus d'indirizzo sono decodificati da questo circuito, che non è un decodificatore in assoluto.

Un'ultima tecnica di decodifica è simile a quella usata nella Figura 17-8 nel Capitolo 17. I bit di indirizzo più elevato sono inseriti in un gate NAND 7430 a otto ingressi, che decodifica gli otto bit in un solo stato con un unico livello logico 0. Per esempio, se i bit da A8 ad A15 sono inseriti in un gate 74L30, verrà prodotta un'uscita di livello logico 0 solo quando A8 = A9 = A10 = A11 = A12 = A13 = A14 = A15 = 1. Questa uscita può essere poi usata per abilitare altri chip di decodifica quali il 74154 o il 7442.

## I/O MEMORY MAPPED: USO DEL BIT D'INDIRIZZO A-15

L'"8080 Microcomputer System User's Manual" della Intel Corporation fornisce interessanti diagrammi che dimostrano come usare il bit del bus d'indirizzo A-15 per distinguere fra memoria e un dispositivo di I/O in mappa di memoria. Nell'I/O tramite l'accumulatore, vengono generati quattro segnali di controllo o dal chip 8228 (Figura 21-3) o da circuiti equivalenti: $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$ e $\overline{OUT}$. Questi segnali vi permettono di distinguere fra una locazione di memoria e un dispositivo di I/O. Nell'I/O memory mapped, si può osservare, come mostra la Figura 21-4, che solo $\overline{MEMR}$ e $\overline{MEMW}$ vengono usati per indirizzare sia la memoria che i dispositivi di I/O. Nella figura, il bit d'indirizzo A-15 forma un gate con questi due segnali di controllo per produrre due segnali di controllo nuovi, $\overline{MEMIOR}$ e $\overline{MEMIOW}$, che vengono usati solo con dispositivi di I/O.

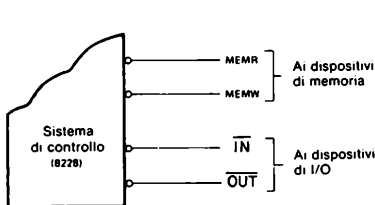



*Figura 21-3.* Segnali di controllo usati nell'I/O tramite accumulatore

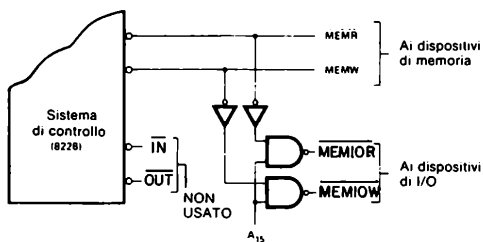



*Figura 21-4.* Segnali di controllo usati in un tipo di I/O memory mapped

Usando il bit d'indirizzo A-15, si suddividono i 65K di memoria in due blocchi di 32K, uno per la memoria e l'altro per i dispositivi di I/O in mappa di memoria. (Vedi Figura 21-5). Al contrario, nel normale I/O tramite l'accumulatore, possono essere indirizzati solo 256 dispositivi di ingresso o 256 dispositivi di uscita, ma la memoria può arrivare al massimo a 65K.

Le tecniche di I/O tramite l'accumulatore e di I/O memory mapped discusse in questo capitolo ed in quelli precedenti non costituiscono tutte le possibilità disponibili. Invece di usare un decodificatore 74154, i bit individuali del codice dispositivo a 8 bit potrebbero essere decodificati direttamente per selezionare sei dispositivi di I/O, ognuno dei quali richiede un codice di selezione porta a 2 bit (Figura 21-6); questo tipo di I/O tramite l'accumulatore è molto utile quando avete soltanto pochi dispositivi di I/O. La stessa tecnica può essere applicata nell'I/O memory mapped, come mostra la Figura 21-7. Nella figura, potete selezionare fino a tredici diversi dispositivi di I/O, ognuno dei quali richiede un codice di selezione porta a 2 bit. Il bit A-15 (Vedi Figura 21-4), è usato per distinguere fra memoria e dispositivo di I/O.

Infine, se volete spostare il blocco di I/O di memoria di 32K in una regione di memoria più piccola, potete decodificare simultaneamente molti dei bit di ordine più elevato sul bus d'indirizzi. Per esempio, se usate una coppia di gate NAND 7420 a 4 ingressi nella Figura 21-4 invece dei gate NAND a 2 ingressi mostrati, potete decodificare i bit d'indirizzo A-13, A-14 e A-15, restringere il blocco di memoria di I/O memory mapped a 8K ed espandere il blocco di memoria a 57K.

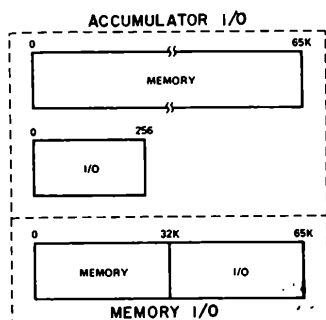



*Figura 21-5.* Confronto del blocco di memoria fra I/O tramite accumulatore e I/O memory mapped.

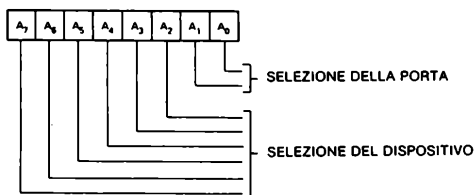



*Figura 21-6.* Esempio di come si usa il codice dispositivo a 8 bit nell'I/O tramite l'accumulatore per indirizzare sei dispositivi, ognuno dei quali richiede un codice porta a 2 bit.

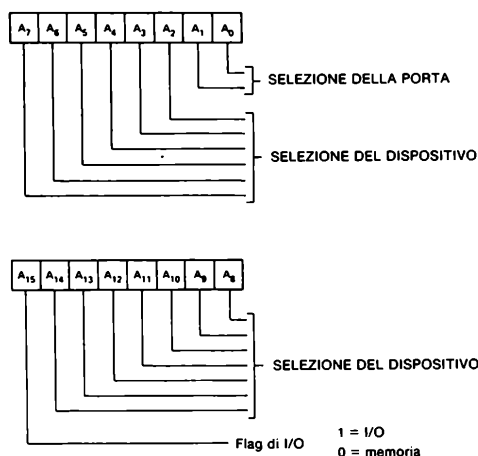



*Figura 21-7.* Esempio di come si usa la parola d'indirizzo di memoria a 16 bit nell'I/O memory mapped per indirizzare tredici diversi dispositivi di I/O, ognuno dei quali richiede un codice porta a 2 bit.

## ISTRUZIONI DI I/O MEMORY MAPPED

Vi sono ventidue istruzioni dell'8080A che vi permettono di trasferire i dati fra i registri interni e i dispositivi di memoria esterni. Questi dispositivi esterni possono essere o memorie a semiconduttore, cioè memorie di lettura-scrittura, ROM, EPROM, ecc., o anche dispositivi di ingresso-uscita che vengono indirizzati come se fossero locazioni di memoria. Compreso in ogni istruzione di riferimento alla memoria vi è un indirizzo di memoria a 16 bit che identifica unicamente un byte di memoria. Questo indirizzo di memoria è contenuto o nella coppia di registri H, B, o D, o anche nei byte <B2> e <B3> dell'istruzione stessa.

In aggiunta alle ventidue istruzioni di trasferimento dati, esistono altre undici istruzioni di riferimento alla memoria. Una di tali istruzioni vi permette di spostare un byte immediato presente in un programma, verso una locazione di memoria. Altre due istruzioni vi permettono di incrementare o decrementare i contenuti di una specifica locazione di memoria. Infine, altre otto istruzioni vi permettono di eseguire operazioni logiche o aritmetiche fra i contenuti di una locazione di memoria e i contenuti dell'accumulatore. Di seguito sono riassunte tutte e trentatre le istruzioni di riferimento alla memoria. Sono suddivise secondo la posizione della parola indirizzo di memoria.

**L'indirizzo della locazione di memoria M è contenuto nella coppia di registri H**

In questo gruppo sono contenute venticinque delle trentatre istruzioni 8080A di riferimento alla memoria. Tutte richiedono che l'indirizzo della locazione di memoria venga memorizzato nella coppia di registri H prima che l'istruzione di riferimento alla memoria sia eseguita.

| | | |
|---|---|---|
| MOV B,M | 106 | Sposta i contenuti della locazione di memoria M nel registro B |
| MOV C,M | 116 | Sposta i contenuti della locazione di memoria M nel registro C |
| MOV D,M | 126 | Sposta i contenuti della locazione di memoria M nel registro D |
| MOV E,M | 136 | Sposta i contenuti della locazione di memoria M nel registro E |
| MOV H,M | 146 | Sposta i contenuti della locazione di memoria M nel registro H |
| MOV L,M | 156 | Sposta i contenuti della locazione di memoria M nel registro L |
| MOV A,M | 176 | Sposta i contenuti della locazione di memoria M nel registro A |
| MOV M,B | 160 | Sposta i contenuti del registro B nella locazione di memoria M |
| MOV M,C | 161 | Sposta i contenuti del registro C nella locazione di memoria M |
| MOV M,D | 162 | Sposta i contenuti del registro D nella locazione di memoria M |
| MOV M,E | 163 | Sposta i contenuti del registro E nella locazione di memoria M |
| MOV M,H | 164 | Sposta i contenuti del registro H nella locazione di memoria M |
| MOV M,L | 165 | Sposta i contenuti del registro L nella locazione di memoria M |
| MOV M,A | 167 | Sposta i contenuti del registro A nella locazione di memoria M |
| MVI M<br><B2> | 066<br><B2> | Sposta il byte immediato <B2> nella locazione di memoria M |

| | | |
|---|---|---|
| INR M | 064 | Incrementa i contenuti della locazione di memoria M |
| DCR M | 065 | Decrementa i contenuti della locazione di memoria M |
| ADD M | 206 | Somma i contenuti della locazione di memoria M ai contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| ADC M | 216 | Somma con riporto i contenuti della locazione di memoria M ai contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| SUB M | 226 | Sottrai con i contenuti della locazione di memoria M dai contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| SBB M | 236 | Sottrai con riporto negativo i contenuti della locazione di memoria M dai contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| ANA M | 246 | AND dei contenuti della locazione di memoria M con i contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| XRA M | 256 | OR esclusivo dei contenuti della locazione di memoria M con i contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| ORA M | 266 | OR dei contenuti della locazione di memoria M con i contenuti dell'accumulatore e memorizza il risultato nell'accumulatore |
| CMP M | 276 | Confronta i contenuti della locazione di memoria M con i contenuti dell'accumulatore. Lascia inalterato l'accumulatore e altera i bit dei flag perchè corrispondano ai risultati dell'operazione di confronto. |

**L'indirizzo della locazione di memoria M è contenuto nella coppia di registri B**

In questo gruppo sono contenute solo due delle trentatre istruzioni di riferimento alla memoria, STAX B e LDAX B.

| | | |
|---|---|---|
| STAX B | 002 | Memorizza i contenuti dell'accumulatore alla locazione di memoria M data dai contenuti della coppia di registri B |
| LDAX B | 012 | Carica l'accumulatore con i contenuti della locazione di memoria M data dai contenuti della coppia di registri B |

**L'indirizzo della locazione di memoria M è contenuto nella coppia di registri D**

| | | |
|---|---|---|
| STAX D | 022 | Memorizza i contenuti dell'accumulatore nella locazione di memoria M data dai contenuti della coppia di registri D |
| LDAX D | 032 | Carica l'accumulatore con i contenuti della locazione di memoria M data dai contenuti della coppia di registri D |

**L'indirizzo della locazione di memoria M è contenuto nel secondo e nel terzo byte di istruzioni**

| | | |
|---|---|---|
| STA<br><B2><br><B3> | 062<br><B2><br><B3> | Memorizza i contenuti dell'accumulatore nella locazione di memoria M definita dai byte di istruzioni <B2> e <B3> |
| LDA<br><B2><br><B3> | 072<br><B2><br><B3> | Carica l'accumulatore con i contenuti della locazione di memoria M definita dai byte di istruzioni <B2> e <B3> |
| SHLD<br><B2><br><B3> | 042<br><B2><br><B3> | Memorizza i contenuti del registro L nella locazione di memoria M definita dai byte di istruzioni <B2> e <B3>; memorizza i contenuti del registro H nella locazione di memoria successiva, M+1. (NOTA: questo è un trasferimento di dati a due byte in una sola istruzione). |
| LHLD<br><B2><br><B3> | 052<br><B2><br><B3> | Carica il registro L con i contenuti della locazione di memoria M definita dai byte di istruzione <B2> e <B3>; carica il registro H con i contenuti della locazione di memoria successiva, M+1. (NOTA: questo è un trasferimento di dati a due byte in una sola istruzione). |

Le istruzioni SHLD e LHLD differiscono dalle altre trentuno istruzioni di riferimento alla memoria per il fatto che vengono trasferiti due byte di dati.

## I CICLI MACCHINA MEMORY READ E MEMORY WRITE

Come con le istruzioni IN e OUT, il microprocessor 8080A ha un ciclo macchina durante il quale avviene il trasferimento dei dati fra la posizione di memoria e i registri interni. Il ciclo macchina è chiamato o MEMORY READ o MEMORY WRITE, e durante tale ciclo avviene quanto segue:

- Viene generato un impulso $\overline{\text{MEMR}}$ o $\overline{\text{MEMW}}$ sul bus di controllo.
- Appare un solo indirizzo di memoria a 16 bit sul bus d'indirizzo.
- Il bus di dati esterno bidirezionale e il bus di dati interno al chip microprocessore vengono aperti per permettere la comunicazione diretta dei dati fra uno dei registri universali interni e il dispositivo di I/O, o di ingresso o di uscita.

L'istruzione SHLD differisce dalle altre per il fatto che vengono eseguiti dall'8080A due successivi cicli macchina MEMORY WRITE. Con l'istruzione LHLD, vengono eseguiti due successivi cicli macchina MEMORY READ. In tutti gli altri casi, viene eseguito solo un ciclo macchina, o MEMORY READ o MEMORY WRITE.

## PRIMO PROGRAMMA

Prendete in considerazione il programma seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 062 | START, STA | Scrivi i contenuti dell'accumulatore nel dispositivo di uscita in mappa di memoria, che ha l'indirizzo di memoria seguente. |
| 001 | 000 | 000 | Byte d'indirizzo LO del dispositivo di uscita. |
| 002 | 200 | 200 | Byte d'indirizzo HI del dispositivo di uscita. |
| 003 | 074 | INR A | Incrementa l'accumulatore. |
| 004 | 303 | JMP | Salto incondizionato alla locazione di memoria START. |
| 005 | 000 | START | Byte d'indirizzo LO di START. |
| 006 | 003 | – | Byte d'indirizzo HI di START. |

In questo programma, abbiamo supposto che non esista memoria nelle locazioni HI = 200 e LO = 000. Questa è un'ottima supposizione per la maggior parte dei microcomputer basati sull'8080A.

Se volete eseguire questo programma secondo la tecnica del passo-passo, dovreste osservare, in successione, i byte seguenti sul bus di dati bidirezionale:

| Byte del bus di dati | Commento |
|---|---|
| 062 | Ciclo macchina FETCH per il codice istruzione STA |
| 000 | Ciclo macchina FETCH per il byte <B2> dell'istruzione STA |
| 200 | Ciclo macchina FETCH per il byte <B3> dell'istruzione STA |
| contenuti dell'accumulatore | Ciclo macchina MEMORY WRITE, durante il quale i contenuti dell'accumulatore vengono messi a disposizione sul bus di dati bidirezionale, l'indirizzo di memoria <B2> e <B3> appare sul bus d'indirizzo, e viene generato un impulso di controllo $\overline{MEMW}$. |
| 074 | Ciclo macchina FETCH per il codice istruzione INR A |
| 303 | Ciclo macchina FETCH per il codice istruzione JMP |
| 000 | Ciclo macchina FETCH per il byte d'indirizzo LO di START |
| 003 | Ciclo macchina FETCH per il byte d'indirizzo HI di START |

Osservate tale informazione sul bus di dati perché (a) tutti i byte istruzioni si muovono sul bus di dati dalla memoria di lettura/scrittura o EPROM al registro istruzioni all'interno del chip 8080A, e (b) i contenuti dell'accumulatore sono messi in uscita sul bus dati durante il quarto ciclo macchina dell'istruzione STA.

Il programma incrementa i contenuti dell'accumulatore durante ogni loop. Inoltre, esso mette in uscita i contenuti dell'accumulatore sul dispositivo di I/O in mappa di memoria, HI = 200 e LO = 000. Si dovrebbe usare il circuito di decodifica mostrato in Figura 21-1.

## ALCUNI CIRCUITI DI INGRESSO/USCITA

I circuiti di ingresso-uscita che usano l'indirizzamento di I/O memory mapped sono identici a quelli mostrati per l'I/O tramite l'accumulatore nel Capitolo N. 20. L'unica differenza è il tipo di impulsi di selezione usati. Vi forniamo parecchi esempi per dimostrarvi la somiglianza e vi rimandiamo poi al Capitolo precedente.

Un circuito di uscita in mappa di memoria, che è basato su di un registro a scorrimento a 8 bit 74198, è mostrato nella Figura 21-3. L'impulso di selezione indirizzi è un impulso $\overline{MEMW}$ codificato per l'indirizzo di memoria HI = 200 e LO = 200, come sarebbe generato dal circuito della Figura 21-1. Un circuito analogo è basato su una coppia di latch tipo D 7475, come mostra la Figura 21-4. L'indirizzo di memoria del latch di uscita è comunque HI = 200 e LO = 001. In questi due circuiti di uscita, vengono usati i display a sette segmenti per due differenti scopi. Per il registro a scorrimento 74198, supponiamo che l'uscita sia una coppia di cifre BCD impaccate, mentre per i chip 7475, supponiamo che l'uscita sia in codice binario a 8 bit, che noi decodifichiamo come tre cifre ottali.

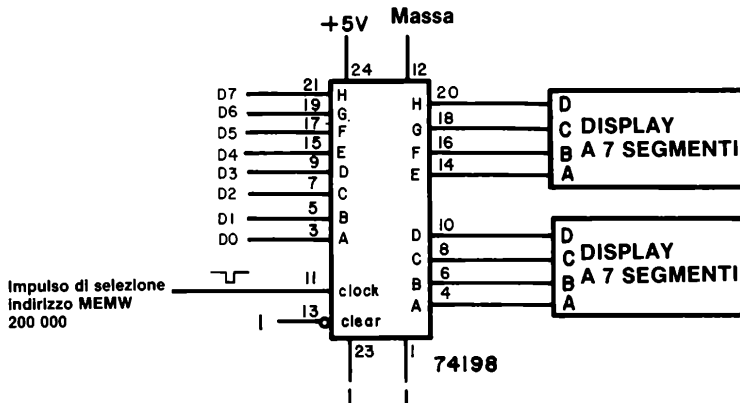



*Figura 21-3.* Circuito di uscita del microcomputer, basato sull'uso della tecnica di I/O memory mapped applicata al registro a scorrimento a 8 bit 74198. L'indirizzo di memoria di questa porta di uscita è di HI = 200 e LO = 000, cioè il bit d'indirizzo A-15 è usato per identificare questo chip come una porta di I/O memory mapped.

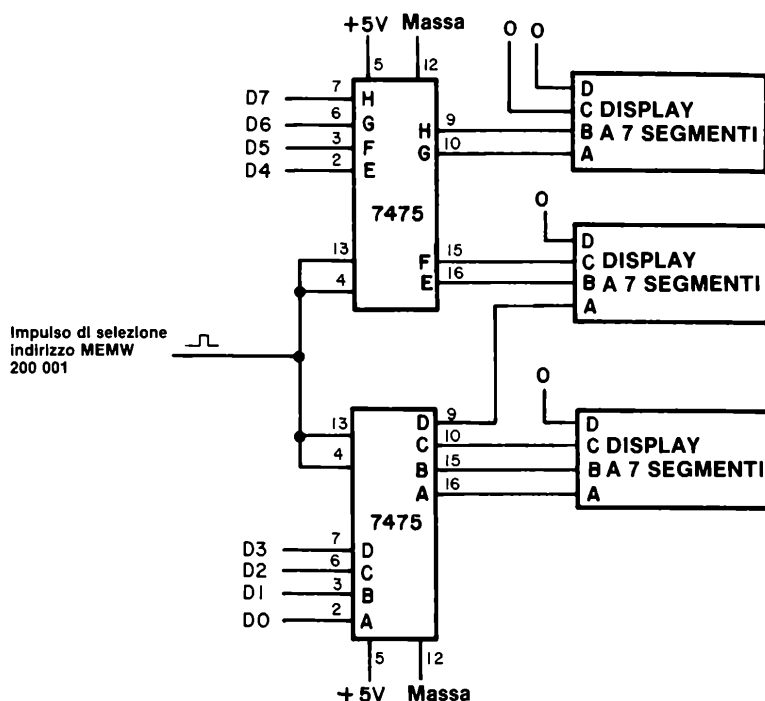



*Figura 21-4.* Circuito di uscita del microcomputer basato sulla tecnica di I/O memory mapped applicata ad una coppia di latch tipo D 7475. L'indirizzo di memoria di questa porta di uscita è HI = 200 e LO = 001.

Il terzo ed ultimo circuito d'ingresso del microcomputer è basato sul latch/buffer a 8 bit 8212. L'impulso di selezione indirizzi è generato dal segnale di controllo $\overline{MEMR}$ e dal bus d'indirizzo a 16 bit, ed ha un indirizzo di memoria di HI = 200 e LO = 002. Per il resto il circuito è identico a quello mostrato nella Figura 20-18.

Dovrebbe essere chiara la rassomiglianza fra queste figure e quelle corrispondenti per l'I/O tramite l'accumulatore nel Capitolo N. 20. Infatti, le Figure 20-18 e 21-5 sono identiche ad eccezione dell'identificazione dell'impulso di selezione. Vi rimandiamo alle Figure 20-6. 20-17 per altri utili circuiti di ingresso-uscita che possono essere adattati all'I/O memory mapped.

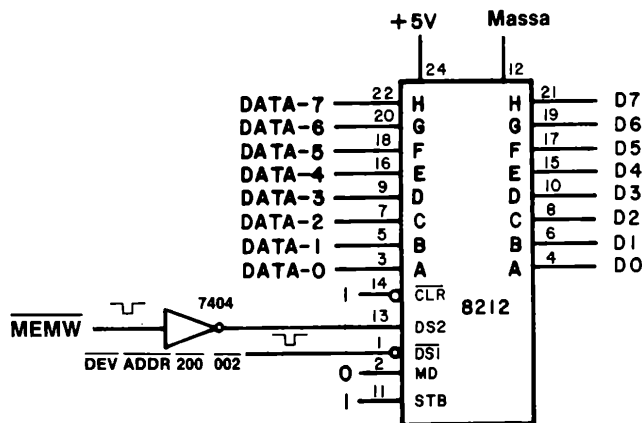



*Figura 21-5.* Circuito d'ingresso del microcomputer basato sulla tecnica di I/O memory mapped applicata ad un latch/buffer a 8 bit 8212. L'indirizzo di memoria di questa porta d'ingresso è HI = 200 e LO = 002.

**SECONDO PROGRAMMA**

Il programma seguente è identico, durante l'esecuzione, al primo programma:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | START, LXI H | Carica la coppia di registri H con due byte seguenti |
| 001 | 000 | 000 | Byte del registro L, il byte d'indirizzo della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 167 | LOOP, MOV M,A | Scrivi i contenuti dell'accumulatore nella locazione di memoria M |
| 004 | 074 | INR A | Incrementa l'accumulatore |
| 005 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

## TERZO PROGRAMMA

Un terzo modo di ottenere il risultato desiderato del primo e del secondo programma è l'uso dell'sitruzione STAX:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 001 | START, LXI B | Carica la coppia di registri B con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro C, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro B, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 002 | LOOP, STAX B | Scrivi i contenuti dell'accumulatore nella locazione di memoria M identificata dai contenuti della coppia di registri B |
| 004 | 074 | INR A | Incrementa l'accumulatore |
| 005 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

Notate che questa volta l'identificazione della locazione di memoria di uscita M è contenuta all'interno della coppia di registri B. Per il resto l'esecuzione del programma è identica a quella del primo e del secondo programma.

## QUARTO PROGRAMMA

La coppia di registri D può essere usata per identificare la locazione di memoria M. Perciò:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 021 | LXI D | Carica la coppia di registri D con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro E, il byte d'indirizzo LO della locazione di memoria M |

| | | | |
|---|---|---|---|
| 002 | 200 | 200 | Byte del registro D, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 022 | LOOP, STAXD | Scrivi i contenuti dell'accumulatore nella locazione di memoria M identificata dai contenuti della coppia di registri D |
| 004 | 074 | INR A | Incrementa l'accumulatore |
| 005 | 303 | JMP | Salto incodizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

Questo programma è essenzialmente uguale al terzo programma. Si dovrebbe osservare che STAX H è l'equivalente di MOV M,A.

## QUINTO PROGRAMMA

I programmi d'ingresso con la tecnica di I/O memory mapped sono semplici come i programmi di uscita suddetti. Consideriamo un sistema nel quale sia le porte d'ingresso che di uscita hanno lo stesso indirizzo di memoria, cioè HI = 200 e LO = 000. Il seguente programma vi permetterà di visualizzare i dati d'ingresso:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 001 | LXI B | Carica la coppia di registri B con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro C, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro B, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 012 | LOOP, LDAXB | Carica l'accumulatore dalla porta di ingresso identificata dai contenuti della coppia di registri B |
| 004 | 002 | STAX B | Scrivi i contenuti dell'accumulatore nella porta di uscita M identificata dai contenuti della coppia di registri B |
| 005 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

Se l'indirizzo di memoria M è contenuto nella coppia di registri D invece che nella coppia di registri B, dovreste sostituire LDAX D e STAX D con i byte di istruzioni a LO = 003 e LO = 004.

## SESTO PROGRAMMA

Il programma seguente, una volta eseguito, dà lo stesso risultato che abbiamo osservato nel quinto programma:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 176 | LOOP, MOV A,M | Carica l'accumulatore con i contenuti della porta di ingresso M |
| 004 | 167 | MOV M,A | Scrivi i contenuti dell'accumulatore nella porta di uscita M |
| 005 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

Come per il quinto programma, le porte di ingresso e di uscita hanno la stessa locazione di memoria, HI = 200 e LO = 000. Ciò che distingue il trasferimento dei dati fra le due porte, che potete vedere nella Figura 21-6, è il modo in cui il bus di dati all'interno dell'8080A opera ed anche l'esistenza dei diversi segnali di controllo, $\overline{MEMR}$ e $\overline{MEMW}$. Vi presentiamo la Figura 21-6 come un circuito che ha valore educativo ma non come un circuito che dovreste cablare in un sistema a interfacce nel microcomputer. Perché no? La risposta è che *non avete bisogno di cablare una porta d'ingresso 8212 allo scopo di visualizzare l'uscita dalla porta di uscita 8212 della Figura 21-6.* Vi raccomandiamo di memorizzare i contenuti in uscita in un registro interno o in una locazione di memoria di lettura/scrittura prima o dopo aver messo in uscita la parola a 8 bit sulla porta di uscita 8212.

Ricordate: nella maggior parte dei casi il vostro obiettivo è di sostituire l'hardware con il software. Usate la vostra memoria di lettura/scrittura per la memorizzazione delle parole di controllo e di altri tipi di informazioni temporali. Non aggiungete circuiti integrati ulteriori *al vostro circuito di interfaccia, a meno che non siano assolutamente necessari.*

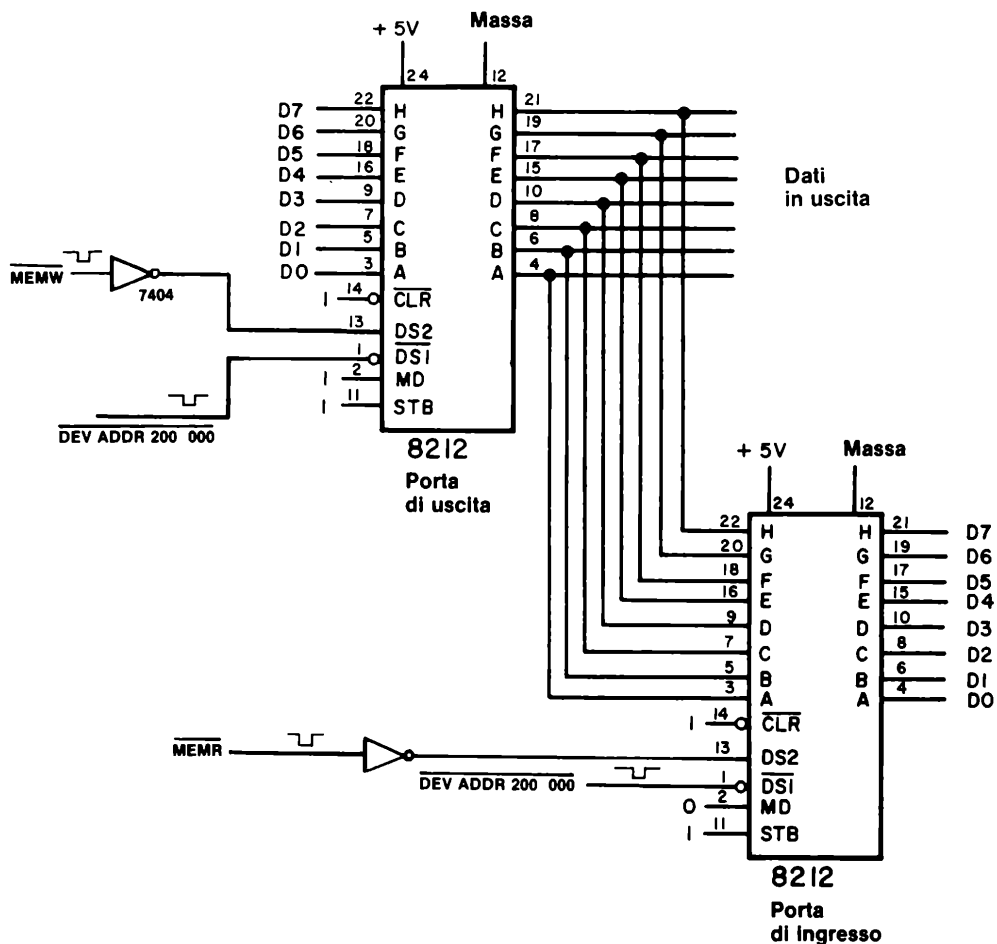



*Figura 21-6.* Circuito di interfaccia di I/O memory mapped dimostra che una porta d'ingresso e una porta di uscita possono avere lo stesso indirizzo di memoria. In generale, identificherete le due porte in questa figura con diversi indirizzi di memoria.

## SETTIMO PROGRAMMA

Non è necessario inserire e mettere in uscita i dati da e verso l'accumulatore, come abbiamo fatto con tutti i programmi precedenti. Per esempio, nel sesto programma, potremmo scambiare i dati con il registro E. Ve lo mostriamo qui di seguito.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |

| | | | |
|---|---|---|---|
| 001 | 001 | 001 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 136 | MOV E,M | Carica il registro E con i contenuti della porta d'ingresso M |
| 004 | 163 | MOV M,E | Scrivi i contenuti del registro del registro E nella porta di uscita M |
| 005 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 006 | 003 | LOOP | Byte d'indirizzo lo di LOOP |
| 007 | 003 | – | Byte d'indirizzo HI di LOOP |

I dati in ingresso con la tecnica di I/O memory mapped possono essere scambiati con i registri B, C, D, o E. Vi raccomandiamo di non usare la coppia di registri H a questo scopo, a meno che non vengano usate le istruzioni SHLD e LHLD.

**OTTAVO PROGRAMMA**

Prendete in considerazione il programma seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro H, il byte d'indirizzo HI delle locazioni di memoria da M a M+3 |
| 003 | 106 | MOV B,M | Carica il registro B con i contenuti della porta d'ingresso M |
| 004 | 054 | INR L | Incrementa il registro L |
| 005 | 116 | MOV C,M | Carica il registro C con i contenuti della porta d'ingresso M+1. |
| 006 | 054 | INR L | Incrementa il registro L |
| 007 | 126 | MOV D,M | Carica il registro D con i contenuti della porta d'ingresso M+2 |
| 010 | 054 | INR L | Incrementa il registro L |
| 011 | 136 | MOV E,M | Carica il registro E con i contenuti della porta d'ingresso M+3 |

Questo programma illustra due degli importanti vantaggi delle tecniche di I/O memory mapped:

1. La facilità con cui si può cambiare il codice dispositivo di I/O.

2. La velocità con cui si possono inserire quattro byte di dati nel chip 8080A.

Accanto a questi vantaggi, vanno soppesati due possibili svantaggi delle tecniche di I/O viste:

1. I circuiti aggiuntivi richiesti per la decodificazione in assoluto dell'indirizzo.

2. La perdita di area di memoria quando questa viene suddivisa in blocchi di memoria e blocchi di I/O memory mapped.

Lo svantaggio numero 2 non è importante per quanto riguarda i piccoli sistemi di microcomputer. Con i grossi sistemi che richiedono una memoria considerevole, vi è un notevole incentivo ad aggiungere decodificatori, in modo che solo una piccolissima porzione di memoria viene decodificata in assoluto in codici di indirizzi di I/O memory mapped.

## INTRODUZIONE AGLI ESPERIMENTI

I semplici esperimenti che seguono illustrano le tecniche di I/O memory mapped. Nel Capitolo N. 22, troverete esperimenti più ampi al riguardo.

| Esperimento N. | Commento |
|---|---|
| 1 | Un semplice circuito di I/O in mappa di memoria che consiste di una coppia di latch 7475 e di una coppia di buffer three-state 8095 (74365). Gli impulsi di selezione indirizzi vengono generati con l'aiuto di un gate NAND a 4 ingrssi 74L20. |
| 2 | Ingresso-uscita in mappa di memoria da e verso l'accumulatore. Dimostra l'uso di istruzioni diverse che possono trasferire i dati fra una porta di I/O in mappa di memoria e l'accumulatore, ad es. STA, LDAX B, STAX B, MOV A,M e MOV M,A. |
| 3 | Uso delle istruzioni INR M, DCR M, e MVI M. Dimostra come si può incrementare una porta di I/O in mappa di memoria. |
| 4 | Uso dell'istruzione ANA M. Dimostra come una porta d'ingresso in mappa di memoria può operare in modo logico direttamente sui contenuti dell'accumulatore. |

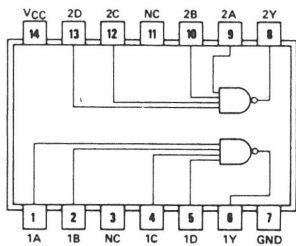
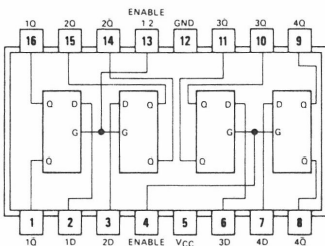
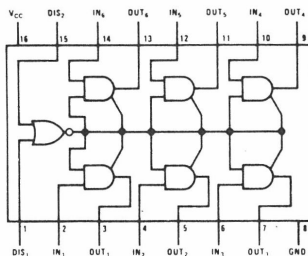
*Le porte di I/O in mappa di memoria, cablate nell'Esperimento N. 1, verranno usate in tutti gli altri esperimenti di questo capitolo.*

## ESPERIMENTO N. 1

## SEMPLICI PORTE DI INGRESSO/USCITA IN MAPPA DI MEMORIA

### Scopo

Lo scopo di questo esperimento è provare il comportamento di un semplice circuito di I/O in mappa di memoria basato su di un buffer three-state 8095 e su di un latch 7475.

### Configurazioni dei pin dei circuiti integrati

### Schemi dei circuiti

+5V Massa

D7 D6 D5 D4

7475

INDICATORI A LED

Impulso di selezione indirizzo

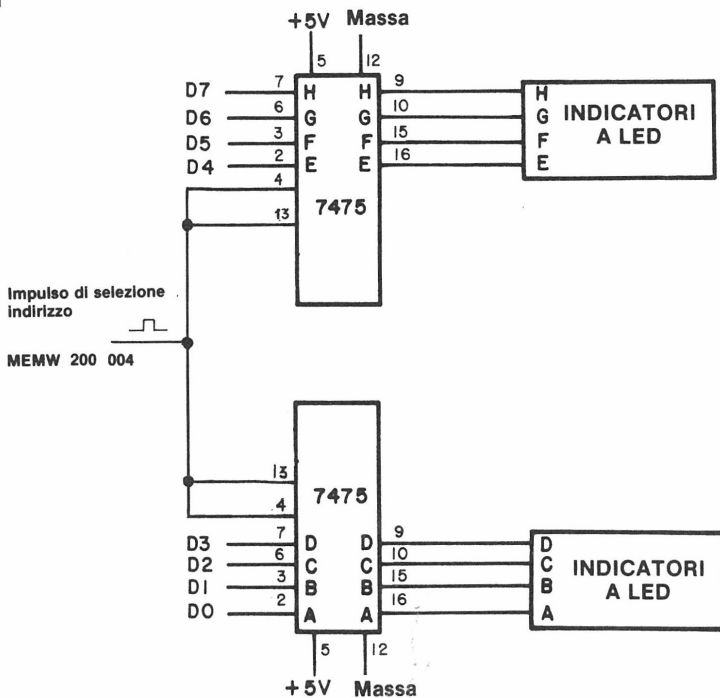
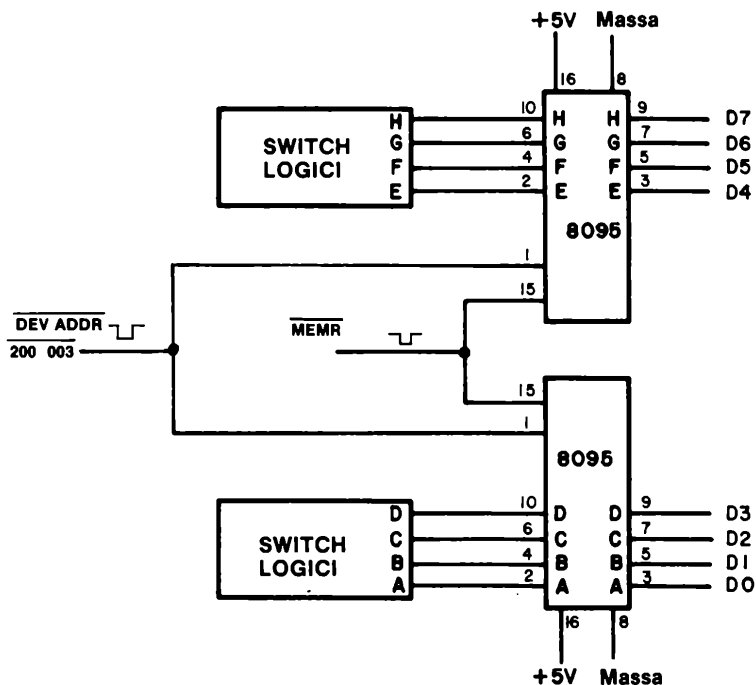
MEMW 200 004

7475

D3 D2 D1 D0

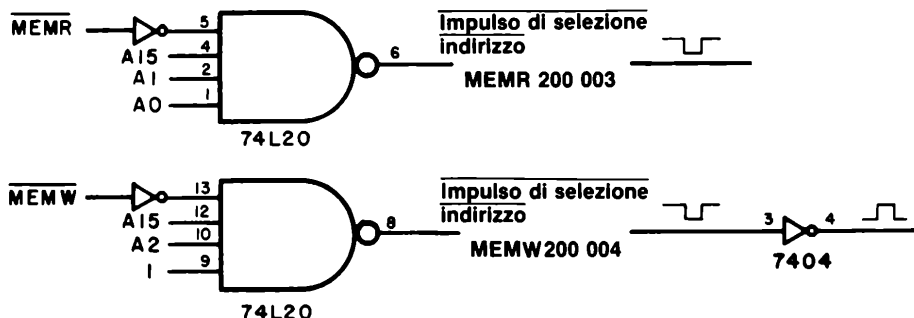
INDICATORI A LED

+5V Massa

21-23

**Programma**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia registri H con i due byte seguenti |
| 001 | 003 | 003 | Byte del registro L |
| 002 | 200 | 200 | Byte del registro H |
| 003 | 106 | START, MOV B,M | Leggi nel registro B i contenuti della porta d'ingresso in mappa di memoria 200 003 |
| 004 | 043 | INX H | Incrementa la coppia di registri H |
| 005 | 160 | MOV M,B | Scrivi i contenuti del registro B nella porta di uscita in mappa di memoria 200 004 |
| 006 | 053 | DCX H | Decrementa la coppia di registri H |
| 007 | 303 | JMP | Salta a START e ripeterla |
| 010 | 003 | START | Byte d'indirizzo LO di START |
| 011 | 003 | – | Byte d'indirizzo HI di START |

**Passo 1**

Cablare il circuito mostrato. Per generare i due impulsi di selezione indirizzi richiesti, usate o un decodificatore o il gate NAND a 4 ingressi 74L20 mostrato qui sotto, che non decodificano in assoluto il bus d'indirizzo a 16 bit e sono perciò solo circuiti dimostrativi. Non dimenticate gli ingressi al chip 74L20: +5 Volt (pin 14) e GND massa (pin 7).

Se avete intenzione di ricavare il bit d'indirizzo A15 direttamente dal chip 8080A, vi raccomandiamo di usare un chip 74L20 e non un 7420 allo scopo di miniminizzare il fan-in. I segnali di controllo $\overline{\text{MEMR}}$ e $\overline{\text{MEMW}}$ devono essere invertiti prima di essere inseriti nel 74L20. L'impulso di scrittura, 200 004, deve essere invertito prima di essere inserito nella coppia di latch 7475.

**Passo 2**

Caricate il programma in memoria. Eseguitelo. Cambiate la disposizione dello switch logico ed osservate l'uscita della porta di uscita 200 004. Che cosa succede?

Abbiamo osservato una corrispondenza uno ad uno fra l'ingresso dello switch logico dei chip 8095 e l'uscita dell'indicatore a LED dai latch 7475.

Quando i dati vengono inseriti nel microcomputer, dove vengono temporaneamente memorizzati?

Nel registro B.

**Passo 3**

Quali cambiamenti è necessario apportare al programma se volete inserire o mettere in uscita dati da e verso il registro E?

Ciò che necessita, è un cambiamento nei byte istruzioni all'indirizzo di memoria LO 003 e 005:

| | | | |
|---|---|---|---|
| 003 | 136 | MOV E,M | Inserisci nel registro E i contenuti della porta d'ingresso in mappa di memoria 200 003 |
| 005 | 163 | MOV M,E | Metti in uscita i contenuti del registro E sulla porta di uscita in mappa di memoria 200 004 |

Fate questi cambiamenti e dimostrate l'operazione del nuovo programma.

**Passo 4**

Che cosa accadrebbe, se cambiaste il byte istruzioni all'indirizzo di memoria LO da 005 a:

| | | | |
|---|---|---|---|
| 005 | 160 | MOV M,B | Metti in uscita i contenuti del registro B sulla porta di uscita in mappa di memoria 200 004 |

ma lasciate invariato a 003 il byte istruzioni all'indirizzo di memoria LO? Cambierebbe l'esecuzione del programma?

Si. Non sarebbe più possibile inserire i dati dello switch logico e poi metterli in uscita sulla porta di uscita in mappa di memoria. Il problema è che tali dati vengono inseriti nel registro E dove non accade più nulla. L'uscita dei dati è dal registro B, che non viene cambiato dal programma modificato. Abbiamo osservato un'uscita di 000 quando abbiamo tentato questo esperimento.

**Passo 5**

Eseguite i seguenti cambiamenti nel programma:

| | | | |
|---|---|---|---|
| 003 | 116 | START, MOV C,M | Leggi nel registro C i contenuti della porta d'ingresso in mappa di memoria 200 003 |
| 004 | 043 | INX H | Incrementa la coppia di registri H |
| 005 | 121 | MOV D,C | Sposta i contenuti del registro C nel registro D |
| 006 | 162 | MOV M,D | Scrivi i contenuti del registro D nella porta di uscita in mappa di memoria 200 004 |
| 007 | 053 | DCX H | Decrementa la coppia di registri H |
| 010 | 303 | JMP | Salta a START e ripeterla |
| 011 | 003 | START | Byte d'indirizzo LO di START |
| 012 | 003 | — | Byte d'indirizzo HI di START |

Eseguite il programma. Che cosa osservate sulla porta di uscita quando cambiate la disposizione dello switch logico?

Abbiamo osservato una corrispondenza fra lo stato logico della porta di uscita e la disposizione dello switch logico.

**Passo 6**

Ora cambiate il byte istruzioni 005 all'indirizzo di memoria LO in un'istruzione NOP:

| 005 | 000 | NOP | Nessuna operazione |
|---|---|---|---|

Eseguite di nuovo il programma. Che cosa succede? Perché?

Non vi è più una corrispondenza fra i dati d'ingresso alla memoria e le porte di uscita. La ragione sta nel fatto che abbiamo eliminato l'istruzione MOV che trasferisce il byte di dati dal registro C al registro D, dal quale tale byte viene messo in uscita.

**Passo 7**

Quali registri si possono usare quando si usa la tecnica di I/O memory mapped? Quali registri sono coinvolti con l'I/O tramite l'accumulatore?

L'I/O memory mapped può usare uno qualunque dei registri universali, compresi A, B, C, D, E, H, o L, come sorgente o destinazione dei dati. L'I/O tramite l'accumulatore è limitato al registro A come sorgente o destinazione dei dati.

*Conservate i circuiti della porta di I/O 7475 e 8095 e passate all'esperimento seguente.*

## ESPERIMENTO N. 2

## INGRESSO/USCITA MEMORY MAPPED DA E VERSO L'ACCUMULATORE

### Scopo

Lo scopo di questo esperimento è provare le varie istruzioni di riferimento alla memoria che trasferiscono i dati fra porte di ingresso-uscita e l'accumulatore.

### Schema del circuito

Usate le porte di ingresso e uscita in mappa di memoria descritte nell'Esperimento N. 1.

### Programma N. 1

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 072 | LOOP, LDA | Carica l'accumulatore dalla porta di ingresso identificata dall'indirizzo di memoria seguente |
| 001 | 003 | 003 | Byte d'indirizzo LO della porta d'ingresso |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta d'ingresso |
| 003 | 062 | STA | Memorizza i contenuti dell'accumulatore nella porta di uscita identificata dall'indirizzo di memoria seguente |
| 004 | 004 | 004 | Byte d'indirizzo LO della porta di uscita |
| 005 | 200 | 200 | Byte d'indirizzo HI della porta di uscita |
| 006 | 303 | JMP | Salto indietro a LOOP |
| 007 | 000 | LOOP | Byte d'indirizzo LO di LOOP |
| 010 | 003 | — | Byte d'indirizzo HI di LOOP |

### Programma N. 2

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 001 | LXI B | Carica la coppia di registri B con i due byte seguenti |
| 001 | 003 | 003 | Byte d'indirizzo LO della porta di ingresso |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta d'ingresso |
| 003 | 012 | LOOP, LDAX B | Carica l'accumulatore dalla porta d'ingresso identificata dai contenuti della coppia di registri B |
| 004 | 003 | INX B | Incrementa la coppia di registri B |

| | | | |
|---|---|---|---|
| 005 | 002 | STAX B | Memorizza i contenuti dell'accumulatore nella porta di uscita identificata dai contenuti attuali della coppia di registri B |
| 006 | 013 | DCX B | Decrementa la coppia di registri B |
| 007 | 303 | JMP | Salto indietro a LOOP |
| 010 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 011 | 003 | – | Byte d'indirizzo HI di LOOP |

**Programma N. 3**

| | | | |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 003 | 003 | Byte d'indirizzo LO della porta d'ingresso |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta d'ingresso |
| 003 | 176 | LOOP, MOV A,M | Carica l'accumulatore dalla porta d'ingresso identificata dai contenuti della coppia i registri B |
| 004 | 043 | INX H | Incrementa la coppia di di registri H |
| 005 | 167 | MOV M,A | Sposta i contenuti dell'accumulatore nella porta di uscita identificata dai contenuti attuali della coppia di registri H |
| 006 | 053 | DCX H | Decrementa la coppia di registri H |
| 007 | 303 | JMP | Salto indietro a LOOP |
| 010 | 003 | LOOP | Byte d'indirizzo LO di LOOP |
| 011 | 003 | – | Byte d'indirizzo HI di LOOP |

**Passo 1**

In questo esperimento, vi vengono forniti tre diversi tipi di istruzioni di riferimento alla memoria che possono essere usate per trasferire i dati fra l'accumulatore e i dispositivi esterni di ingresso-uscita. Sebbene le istruzioni di trasferimento dei dati abbiano codici mnemonici diversi, i Programmi N. 2 e 3 sono simili.

Supponiamo che abbiate già cablato il circuito mostrato nell'Esperimento N. 1 Gli impulsi d'indirizzo di memoria possono essere generati usando i semplici circuiti gate 74L20 forniti nel Passo 1 di questo esperimento. Ricordatevi comunque che la coppia di latch 7475 richiede un impulso di selezione positivo; perciò, l'uscita del chip 7420 deve essere invertita.

**Passo 2**

Caricate ed eseguite il programma N. 1 Cambiate la disposizione dello switch logico alla porta d'ingresso 8095 (74365).

Che cosa osservate sugli indicatori a LED collegati ai due latch 7475, che comprendono la porta di uscita in mappa di memoria?

Dovremmo osservare una corrispondenza uno ad uno fra ingresso in mappa di memoria e dati in uscita in mappa di memoria. La risposta dovrebbe essere "istantanea".

**Passo 3**

Cambiando i byte d'indirizzo 004 e 005 all'indirizzo di memoria LO, vi sarebbe possibile mettere in uscita i dati in ingresso in mappa di memoria su una delle tre porte di uscita dell'indicatore a LED sul microcomputer MMD-1? Spiegate la vostra risposta.

Non sarebbe possibile convertire nessuna delle porte di uscita sul microcomputer MMD-1 in porte di uscita in mappa di memoria modificando semplicemente una coppia di byte d'indirizzo di memoria nel programma N. 1. Le tre porte di uscita sull'MMD-1 fanno parte dell'hardware, come le porte di uscita nell'accumulatore. Se volete convertirle in porte di uscita in mappa di memoria, dovreste fare un certo numero di cambiamenti nel cablaggio sulla piastra del circuito stampato. Inoltre, dovreste apportare delle modifiche anche al programma KEX. Vi suggeriamo di non farlo.

Quello che vorremmo precisare è che servono sia l'hardware che il software per determinare la natura di una porta di ingresso-uscita, cioè se è una porta di I/O in mappa di memoria o una porta di I/O tramite l'accumulatore.

**Passo 4**

Caricate ed eseguite il programma N. 2. Cambiate la disposizione dello switch logico sulla porta d'ingresso in mappa di memoria e notate la corrispondenza fra tale disposizione e l'uscita della coppia di latch 7475.

Perchè le istruzioni INX B e DCX B sono necessarie in questo programma?

La porta d'ingresso in mappa di memoria ha un indirizzo di 200 003, mentre la porta di uscita ha un indirizzo di 200 004. Usiamo le istruzioni INX B e DCX B per cambiare l'indirizzo esistente nella coppia delle istruzioni LDAX B o STAX B. In questo modo, siamo in grado di indirizzare entrambe le porte. L'I/O memory mapped ci permette di usare istruzioni che possono modificare un indirizzo di memoria. Quest'ultimo è difficile da implementare con l'I/O tramite l'accumulatore.

**Passo 5**

Caricate ed esguite il programma N. 3.

Cambiate ancora la dis!osizione dello switch logico alla porta d'ingresso in mappa di memoria e notate la corrispondenza fra tale disposizione e l'uscita che appare sugli otto indicatori a LED.

Che differenza osserverete nell'esecuzione di questo programma confrontandolo con l'esecuzione dei programmi 1 e 2?

Non dovremmo osservare nessuna differenza.

**Passo 6**

Parlate delle differenze e delle somiglianze delle istruzioni LDAX B e MOV A,M.

Entrambe queste istruzioni sono simili nel fatto che l'indirizzo di memoria è contenuto in una coppia di registri e che è questo indirizzo che specifica l'indirizzo di memoria della porta d'ingresso in mappa di memoria. La sola differenza fra le due istruzioni è l'identità della coppia di registri. Per LDAX B, la coppia di registri B contiene l'indirizzo, mentre per MOV A,M è la coppia di registri H a contenere l'indirizzo.

**Passo 7**

Parlate delle differenze e somiglianze fra le istruzioni STAX B e MOV M,A.

Entrambe queste istruzioni sono simili nel fatto che l'indirizzo di memoria è contenuto in una coppia di registri e che è questo indirizzo che specifica l'indirizzo di memoria della porta di uscita in mappa di memoria. La sola differenza fra le due istruzioni è l'identità della coppia di registri. Per STAX B, la coppia di registri B contiene l'indirizzo, mentre per MOV M,A è la coppia di registri H a contenere l'indirizzo.

**Passo 8**

Perchè potreste preferire l'uso di un'istruzione STA o LDA piuttosto che di un'istruzione STAX, LDAX, MOV A,M o MOV M,A?

Specificando un indirizzo di memoria come una coppia di byte d'indirizzo, eliminate la necessità di usare una coppia di registri. Esistono solo tre coppie di registri nel chip 8080A, per cui usatele saggiamente.

*Conservate i vostri circuiti di uscita 7475 e di ingresso 8095 (74365) per i due esperimenti successivi.*

## ESPERIMENTO N. 3

## USO DELLE ISTRUZIONI INR M, DCR M, E MVI M

**Scopo**

Lo scopo di questo esperimento è provare il comportamento delle istruzioni INR M, DCR M, MVI M su di una tipica porta di uscita in mappa di memoria.

**Schema del circuito**

Usate la porta di uscita descritta nell'esperimento N. 1

**Programma N. 1**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 004 | 004 | Byte d'indirizzo LO della porta di uscita |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta di uscita |
| 003 | 066 | MVI M | Sposta sulla porta di uscita il byte seguente identificato dai contenuti della coppia di registri H |
| 004 | 111 | 111 | Byte di dati immediato da mettere in uscita |
| 005 | 166 | HLT | Alt |

**Programma N. 2**

| | | | |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 004 | 004 | Byte d'indirizzo LO della porta di uscita |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta di uscita |
| 003 | 066 | MVI M | Sposta il byte seguente sulla porta di uscita identificata dai contenuti della coppia di registri H |
| 004 | 111 | 111 | Byte di dati immediato da mettere in uscita |
| 005 | 064 | INR M | Incrementa la porta di uscita |
| 006 | 166 | HLT | Alt |

**Passo 1**

Cablate la porta di uscita in mappa di memoria, che consiste di due latch 7475, come descritto nell'esperimento N. 1.

**Passo 2**

Caricate ed eseguite il programma N. 1. Che cosa osservate sull'indicatore a LED della porta di uscita?

Dovreste osservare il byte di uscita in codice ottale, 111.

**Passo 3**

Cambiate i valori del byte di dati a LO = 004 ed eseguite di nuovo il programma. Per esempio, provate il byte di dati 333. Che cosa osservate ora sulla porta di uscita?

Dovreste osservare un byte di uscita di 333.

Se ripeteste questo processo con altri byte di dati, concludereste che vi è una corrispondenza uno ad uno fra il byte di dati immediato a LO = 004 e i dati in uscita sulla porta di uscita una volta che il programma è stato eseguito. Quando si usa un'istruzione MVI M, MOV r,M o MOV M, r, l'indirizzo della locazione di memoria deve essere memorizzato nei registri H e L. Essi sono caricati all'inzio del programma con un'istruzione LXI H.

**Passo 4**

Caricate ed eseguite il programma N. 2. Che cosa osservate sulla porta di uscita? Che cosa vi aspettavate, il byte a LO = 004?

Osserviamo un byte di uscita di 000. Voi dovreste probabilmente osservare la stessa cosa. Inizialmente ci aspettiamo di osservare il byte di uscita 112. Le ragioni per cui non abbiamo osservato un tale risultato sono discusse nel prossimo passo.

**Passo 5**

Quali operazioni deve eseguire il microprocessor allo scopo di eseguire con successo un'istruzione INR M?

Per prima cosa, i contenuti attuali della locazione di memoria M devono essere inseriti nell'8080A. Poi, devono essere incrementati di uno. Infine, il valore incrementato deve essere rimesso in uscita sulla locazione di memoria M. In altre parole, con una tipica locazione di memoria di lettura/scrittura, l'INR M esegue sia una lettura che una scrittura.

**Passo 6**

Nell'I/O memory mapped, allo scopo di eseguire con successo un'istruzione INR M, quali condizioni devono sussistere alla porta di I/O?

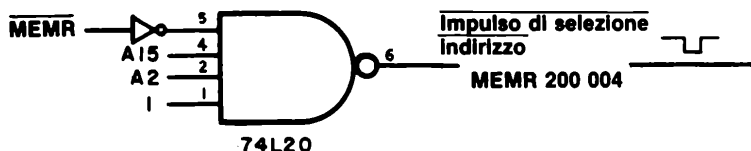
La porta deve essere simile a quella mostrata nella Figura 21-6, cioè dovete essere in grado di leggere dalla e di scrivere nella porta. Una tale condizione esiste già naturalmente nella memoria di lettura/scrittura, ma può non esistere nei circuiti a interfaccia di I/O in mappa di memoria.

**Passo 7**

Perchè abbiamo osservato 000 come byte di uscita nel passo 4 di questo esperimento?

Abbiamo tentato di leggere da una locazione di memoria inesistente, 200 004 e di inserire il byte di dati per difetto 377 (dovuto allo stato floating del bus dati), che è stato incrementato e scritto nella porta di uscita in mappa di memoria come 000. Le istruzioni INR M e DCR M non avvengono su di un byte di dati.

**Passo 8**

Se la porta d'ingresso three-state 8095 (74365) è ancora collegata, cambiate la decodifica dell'indirizzo con la seguente:

Ora la porta d'ingresso e la porta di uscita hanno lo stesso indirizzo. Cambiate i byte istruzioni a LO = 003 e LO = 004 nel programma N. 2 in NOOP,000. Posizionate un valore sui switch logici ed eseguite il programma N. 2. Quale uscita appare sull'indicatore a LED?

L'uscita dell'indicatore a LED dovrebbe essere il vostro posizionamento binario a 8 bit incrementato di 1.

## ESPERIMENTO N. 4

## USO DELL'ISTRUZIONE ANA M

**Scopo**

Lo scopo di questo esperimento è dimostrare l'esecuzione di un'operazione AND fra una porta d'ingresso in mappa di memoria e l'accumulatore.

**Schema del circuito**

Usate la porta d'ingresso descritta nell'esperimento N. 1 *ricablata per l'indirizzo 200 003.*

**Programma**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 003 | 003 | Byte d'indirizzo LO della porta d'ingresso |
| 002 | 200 | 200 | Byte d'indirizzo HI della porta d'ingresso |
| 003 | 076 | TEST, MVI A | Sposta in modo immediato il byte nell'accumulatore |
| 004 | 001 | 001 | Maschera del byte |
| 005 | 246 | ANA M | AND dei contenuti della porta d'ingresso in mappa di memoria con i contenuti dell'accumulatore |
| 006 | 312 | JZ | Se il risultato è zero, salta indietro a TEST. Altrimenti, passa all'istruzione successiva |
| 007 | 003 | TEST | Byte d'indirizzo LO di TEST |
| 010 | 003 | – | Byte d'indirizzo HI di TEST |
| 011 | 323 | OUT | Il bit di flag è a livello logico 1. Metterlo in uscita sulla porta di uscita seguente |
| 012 | 000 | 000 | Porta di uscita 000 |
| 013 | 166 | HLT | Alt |

**Passo 1**

Cablate il circuito d'ingresso in mappa di memoria mostrato nell'esperimento N. 1 se non è già cablato sul vostro breadboard.

**Passo 2**

Caricate ed eseguite il programma N. 1 con lo switch logcio A a livello logico 0. Ora posizionate lo switch logico sul livello logico 1. Che cosa succede sulla porta di uscita 000?

Il bit D0 si accende.

**Passo 3**

Cambiate il byte di maschera a LO = 004 in uno dei seguenti: 200, 100, 040, 020, 010, 004, o 002. Posizionate tutti e otto gli switch logici sul livello logico 0. Eseguite di nuovo il programma e provate ogni switch logico fino a che scoprite quello che non è mascherato. Come fate a sapere quando avete trovato quello giusto?

Il bit che corrisponde al bit non mascherato si accende alla porta di uscita 000

**Passo 4**

Potete provare anche le altre istruzioni di riferimento alla memoria, comprese:

| | |
|---|---|
| 206 | ADD M |
| 216 | ADC M |
| 226 | SUB M |
| 236 | SBB M |
| 256 | XRA M |
| 266 | ORA M |
| 276 | CMP M |

L'istruzione CMP M non coinvolge nessun dato o i contenuti del registro dell'accumulatore. Essa setta e azzera i flag.

**Passo 5**

Perchè sarebbe utile essere in grado di eseguire un'operazione aritmetica o logica fra una porta d'ingresso in mappa di memoria e i contenuti dell'accumulatore?

Potrebbe essere utile se voi desideraste posizionare esternamente un byte maschera per un'operazione ANA M usando un set di otto switch logici.

## DOMANDE RIEPILOGATIVE

Le seguenti domande vi aiuteranno a rivedere le tecniche di I/O memory mapped:

1. Che cosa si intende con il termine "ingresso/uscita memory mapped"?

2. Fate un elenco delle differenze fra I/O tramite l'accumulatore e I/O memory mapped. Per esempio, quali segnali di controllo vengono usati, quali istruzioni, e quali registri, nelle due tecniche di I/O?

3. Che cosa si intende con il termine "decodificazione in assoluto" del bus d'indirizzo nell'I/O memory mapped?

4. Perchè la decodificazione in assoluto del bus d'indirizzo nell'I/O memory mapped è importante?

5. In questo capitolo ed in quelli precedenti, abbiamo usato i termini "impulso di selezione dispositivo" e "impulso di selezione indirizzi". Qual'è la differenza fra questi due termini?

6. Abbiamo sentito dire che la ragione per cui si usano le tecniche di I/O memory mapped è il fatto che permettono di trasferire i dati fra più dispositivi che non le tecniche di I/O tramite l'accumulatore. Siete d'accordo? Se sì, perché? Se no, perché?

**RISPOSTE**

1. Il termine "ingresso-uscita memory mapped viene associato con i sistemi di microcomputer a 8 bit 6800, 8080A, ed altri. Le istruzioni di I/O sono istruzioni di riferimento alla memoria e il trasferimento dei dati avviene, nel caso del chip 8080A, fra il dispositivo di I/O e uno qualunque dei registri universali all'interno del chip 8080A.

2. Nell'I/O tramite l'accumulatore, i segnali di controllo sono $\overline{IN}$ e $\overline{OUT}$, mentre nell'I/O memory mapped i segnali di controllo sono $\overline{MEMR}$ e $\overline{MEMW}$. L'I/O tramite l'accumulatore impiega solo due istruzioni 8080A, IN e OUT. L'I/O memory mapped impiega qualunque istruzione di riferimento alla memoria, es. MOV r,M, MOV M,r, MVI M, STAX rp, LDAX rp, ANA M, ORA M, ADD M, ed altre. Nell'I/O tramite l'accumulatore, il trasferimento dei dati avviene fra il dispositivo di I/O e il registro dell'accumulatore. Nell'I/O memory mapped, il trasferimento dei dati avviene fra il dispositivo di I/O e uno qualunque dei registri universali, quali B, C, D, E, H, L, o l'accumulatore.

3. Tutti e sedici i bit del bus d'indirizzo sono decodificati usando una rete di decodifica adatta in modo che ogni dispositivo di I/O in mappa di memoria è identificato in modo unico e non può essere confuso con una locazione di memoria nella memoria di lettura/scrittura o EPROM o indirizzato accidentalmente durante l'esecuzione del programma

4. Per evitare l'indirizzamento accidentale di un dispositivo di I/O in mappa di memoria durante l'esecuzione del programma. Per distinguere chiaramente fra una locazione di memoria e un dispositivo di I/O in mappa di memoria.

5. Un impulso di selezione dispositivo è generato quando viene applicato il codice dispositivo a 8 bit dal bus d'indirizzo oppure $\overline{IN}$ o $\overline{OUT}$, che sono segnali di controllo, ad un circuito di decodifica adatto. Un impulso di selezione indirizzi è generato quando viene applicato il bus d'indirizzo a 16 bit oppure $\overline{MEMR}$ o $\overline{MEMW}$, che sono segnali di controllo di riferimento alla memoria, ad un circuito di decodifica adatto.

6. No, non siamo d'accordo. Nella maggior parte dei casi, 256 diversi dispositivi d'ingresso e 256 diversi dispositivi di uscita sono più che adeguati per un sistema di microcomputer basato sull'8080A. Una ragione più valida per usare le tecniche di I/O memory mapped è il permettere il trasferimento diretto dei dati fra il dispositivo di I/O e tutti i registri universali all'interno del chip 8080A. Inoltre, i contenuti di una porta d'ingresso in mappa di memoria possono essere aggiunti, o sottratti, o confrontati o operati in maniera logica sui contenuti dell'accumulatore.

# CAPITOLO N. 22

# L'INGRESSO/USCITA DEL MICROCOMPUTER: ALCUNI ESEMPI

## INTRODUZIONE

Uno degli usi più importanti del microcomputer in un laboratorio è come sistema di acquisizione dati. Questo Capitolo esamina i principi dell'acquisizione dati e fornisce un numero di esempi sperimentali dei circuiti di acquisizione dati.

## OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Dare la definizione di "acquisizione dati" e fare le considerazioni più importanti per quanto riguarda lo sviluppo di un sistema di acquisizione dati.
- Descrivere vari metodi per generare ritardi di tempo.
- Realizzare un semplice circuito di acquisizione dati.
- Interfacciare un convertitore digitale analogico AD7522.

## L'ACQUISIZIONE DATI CON UN MICROCOMPUTER 8080

*Un sistema di acquisizione dati* può essere definito nel modo seguente:

| | |
|---|---|
| *Sistema di acquisizione dati* | Uno strumento che scandisce automaticamente i dati prodotti da un altro strumento, o elabora e registra le letture dei dati da usare in seguito. |

Dovrebbe essere chiaro il fatto che un microcomputer può costituire un sistema di acquisizione dati. I dati provenienti da uno strumento possono essere inseriti nell'accumulatore e poi memorizzati. In un secondo tempo, questa informazione memorizzata può essere letta in modi diversi. L'acquisizione dati diventerà un'applicazione comune per i microcomputer.

Forse le domande più importanti che vi dovreste porre quando decidete di acquisire dati da uno strumento sono le seguenti: (1) Quanti canali volete acquisire? (2) Quanto tempo ci vorrà per acquisire tutti questi canali? (3) Quante informazioni digitali sono contenute in un solo canale? (4) Che cosa volete farne dei dati una volta acquisiti? (5) Avete bisogno di una memorizzazione a breve o a lungo termine? Tratteremo ora tutti questi temi.

### QUANTI CANALI DATI?

Il numero di canali che volete acquisire e il tempo che vi occorre per memorizzarli, indicheranno il tipo di dispositivo di memoria richiesto. Se avete bisogno di acquisire un milione di canali di dati a quattro cifre BCD, vi servirà una capacità di memoria di sedici milioni di bit e sarà quindi necessario un nastro magnetico o un disco magnetico. Invece, se avete bisogno di acquisire cento canali, contenenti ognuno solo quattro cifre BCD, e di memorizzare i dati per parecchie ore, sono necessari solo 1600 bit di memoria. Un semplice piastra di memoria di lettura/scrittura andrebbe benissimo per un'applicazione di questo tipo. Se avete bisogno di memorizzare più di 4.000 bit di dati, vi raccomandiamo di usare qualche tipo di nastro magnetico, come un nastro a cassetta o un disco.

### MEMORIZZAZIONE A BREVE O A LUNGO TERMINE?

In genere, la memoria di lettura/scrittura non è adatta per la memorizzazione di dati a lungo termine: essa è infatti volatile; se vi è un'interruzione di tensione, tutti i dati verrano perduti. La memoria a nuclei non è volatile, ma d'altra parte è relativamente costosa e di solito non adatta alla memorizzazione di dati a lungo termine, a meno che la quantità di dati sia limitata. I migliori dispositivi per la memorizzazione di dati sono, come abbiamo detto in precedenza, i nastri a cassetta e i dischi flessibili (Floppy-disk). Una cassetta a nastro di buona qualità può memorizzare fino a 500.000 bit di informazioni su di una sola cassetta. I costi hardware per i Floppy-disk diminuiscono di anno in anno. Lo sviluppo di interfacce LSI molto sofisticate per questi dischi, dovrebbe ridurre ancora di più i costi. Tali chip saranno presto messi a disposizione dalla Intel, Texas Instruments Incorporated, dalla Motorola e dalla National Semiconductor.

Una tecnica di memorizzazione a lungo termine non costosa è costituita dall'uso di un nastro di carta perforato. Vorremmo puntualizzare comunque che occorre parecchio tempo per perforare tale nastro, nonchè leggerlo nel computer. Ad una velocità della telescrivente di 10 caratteri ASCII al secondo, occorrono almeno sette minuti per leggere o perforare 4096 byte di programma o dati.

## QUANTE INFORMAZIONI IN UN SOLO CANALE?

Un tipico canale dati fornisce di solito un numero a tre o quattro cifre BCD, costituito dal valore decimale, o range, e dal segno. Di solito, il valore decimale o il range viene fissato, e il segno è positivo, ma non è sempre così. I nuovi dispositivi digitali vanno incorporando sempre di più una capacità di autorange, il che significa che lo strumento digitale decide dove mettere il valore decimale.

Prendiamo un canale che fornisca almeno sedici bit di informazioni digitali. Per ottenere la capacità totale di memoria richiesta, moltiplicate sedici per il numero di canali. Perciò, per cento canali, sarebbero necessari 1600 bit di memoria di lettura/scrittura. I frequenzimetri hanno molti più bit per canale. Per esempio, un frequenzimetro a sette cifre ha almeno 28 bit per canale.

## CHE COSA FARE DEI DATI ACQUISITI?

Alcuni dei dati acquisiti sono solo dati "originali" che devono essere manipolati e interpretati allo scopo di produrre un risultato finale utile. Un esempio potrebbe essere la conversione di una tensione digitale. In questi casi, i dati acquisiti avranno bisogno di essere elaborati matematicamente, e tale operazione dovrebbe essere eseguita subito dopo che i dati sono stati ottenuti. Nel caso di dati che richiedono di essere ulteriormente elaborati in modo matematico, vi raccomandiamo di tenerli nella forma digitale fino a che non possano essere elaborati. Memorie di lettura/scrittura, nastri magnetici, dischi magnetici, sono tutti adatti a tale scopo. Stampare i dati è una forma di memorizzazione a lungo termine. Sicuramente è il tipo di memorizzazione a lungo termine meno costoso, ma in compenso dovete consumare tempo per riconvertirla in segnali digitali.

## QUANTI CANALI AL SECONDO?

Questo è un punto fondamentale per tutte le operazioni di acquisizione dati. I dati possono, ad esempio, (a) apparire molto lentamente ed occupare dei periodi molto lunghi di tempo, anche un giorno, per essere acquisiti, o (b) apparire molto rapidamente ed occupare solo dei millisecondi per l'acquisizione da centinaia di canali. Entrambi gli estremi nelle velocità di acquisizione dati puntano sul bisogno di tecniche automatiche per acquisire i dati stessi, quali l'uso di un sistema di acquisizione dati basato su un microcomputer. Dato che i microcomputer diminuiscono di prezzo, molti strumenti di laboratorio acquisiranno automaticamente i dati per mezzo di microcomputer incorporati. Si useranno ancora i registratori a carta, ma non sarà necessario che siano della qualità richiesta. Un maggior uso dei registratori a carta nel futuro permetterà agli occhi di "integrare" visivamente un blocco di dati per rilevare la curvatura, la linearità, ecc.

## PRIMO PROGRAMMA: ACQUISIZIONE DI 64 CANALI DI DATI A 8 BIT

Come dimostrazione del concetto di acquisizione dati, vorremmo presentarvi un programma che fa sì che possiate acquisire 64 canali con dati a 8 bit, con la velocità con cui il microcomputer li può inserire e memorizzare. Come esempio di uno "strumento", supponiamo che acquisiate i dati dalla coppia di contatori 7490 mostrati nella Figura 22-1. La domanda a cui vorremmo rispondere è: qual'è il tempo minimo richiesto per acquisire 64 canali a otto bit dalla coppia di contatori?

Il programma, che è un esempio dell'uso delle tecniche di I/O tramite l'accumulatore, è il seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Cicli clock | Descrizione |
|---|---|---|---|---|
| 000 | 041 | LXI H | 10 | Carica la coppia di registri H con i due byte seguenti |
| 001 | 100 | 100 | – | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 003 | 003 | – | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 006 | MVI B | 7 | Sposta il byte seguente nel registro B |
| 004 | 100 | 100 | – | Numero di canali che saranno acquisiti dal microcomputer, cioè 64 |
| 005 | 333 | LOOP, IN | 10 | Inserisci i dati dalla coppia di contatori a decade 7490 |
| 006 | 003 | 003 | – | Codice dispositivo per il buffer d'ingresso della Figura 22-1 |
| 007 | 167 | MOV M,A | 7 | Sposta i contenuti dell'accumulatore nella locazione di memoria M indirizzata dai contenuti della coppia di registri H |
| 010 | 043 | INX H | 5 | Incrementa la coppia di registri H |
| 011 | 005 | DCR B | 5 | Decrementa il registro B |
| 012 | 302 | JNZ | 10 | Se il registro B non è uguale a 000, salta a LOOP; altrimenti, ignora questa istruzione e passa alla successiva |
| 013 | 005 | LOOP | – | Byte d'indirizzo LO di LOOP |
| 014 | 003 | – | – | Byte d'indirizzo HI di LOOP |
| 015 | 166 | HLT | | Alt |

Il loop da LO = 005 a LO = 014 è eseguito 64 volte prima che il microcomputer si arresti. Durante ogni LOOP, occorrono 37 cicli clock. Perciò, il tempo richiesto per acquisire 64 canali è 64 per 37 volte il tempo per ogni ciclo del microcomputer 8080A. Per un microcomputer che opera a 2 MHz, il tempo totale è di 1,184 millisecondi. A 18.5 $\mu$s per ogni canale a 8 bit, un microcomputer a 2 MHz può acquisire circa 54.000 byte al secondo, il che è una notevole quantità di informazioni.

Se il clock della Figura 22-1 operasse a una frequenza di 1 Hz, memorizzereste uno o due valori in tutte le 64 locazioni di memoria. Il modo giusto di eseguire l'esperimento suddetto è usare un ingresso clock che ha una frequenza di almeno 20 kHz. Abbiamo ottenuto buoni risultati usando un clock che aveva una frequenza di 90 kHz.

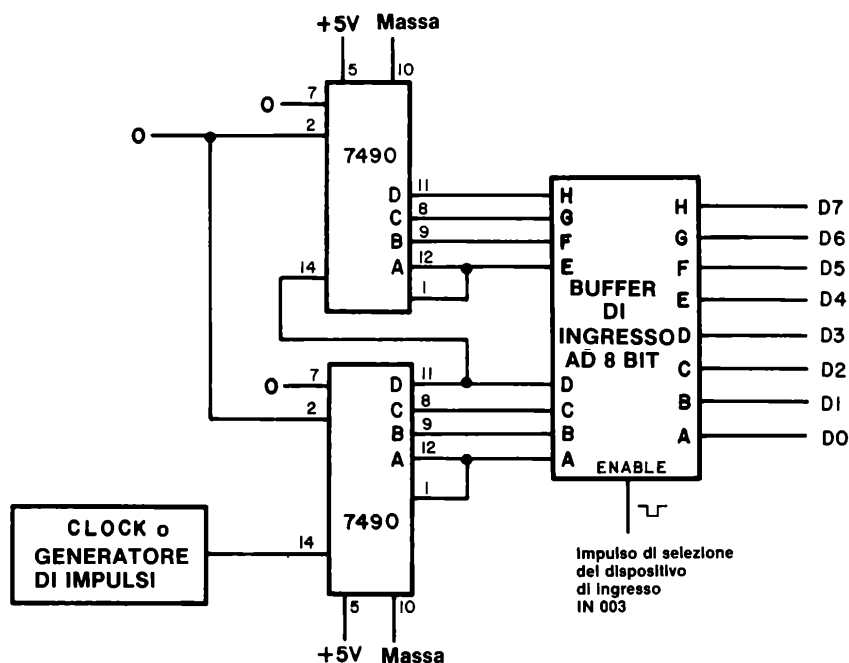



*Figura 22-1.* Semplice circuito di acquisizione dati che impiega una coppia di contatori a decade in cascata 7490.

## SECONDO PROGRAMMA: ACQUISIZIONE LENTA DI CANALI DATI

Non avete bisogno spesso di acquisire dati ad una velocità di 54.000 canali al secondo. Una situazione più comune è quella di una velocità di acquisizione dati di un byte al secondo. L'unico cambiamento richiesto nel Primo Programma è l'inserimento di un loop di ritardo, che ha una durata di circa un secondo. Perciò:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte di dati seguenti |
| 001 | 100 | 100 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 003 | 003 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 006 | MVI B | Carica il registro B con il byte seguente |

| | | | |
|---|---|---|---|
| 004 | 100 | 100 | Numero di canali che saranno acquisiti dal microcomputer, cioè 64 |
| 005 | 333 | LOOP2, IN | Inserisci i dati dalla coppia di contatori a decade 7490 della Figura 22-1 |
| 006 | 003 | 003 | Codice dispositivo per il buffer d'ingresso mostrato nella Figura 22-1 |
| 007 | 167 | MOV M,A | Sposta i contenuti dell'accumulatore nella locazione di memoria M indirizzata dai contenuti della coppia di registri H |
| 010 | 043 | INX H | Incrementa la coppia di registri H |
| 011 | 016 | MVI C | Carica il registro C con il byte di timing seguente, che determina il numero di loop di ritardo di 10 ms |
| 012 | 144 | 144 | Byte di timing per il registro C, che corrisponde a 100 loop |
| 013 | 315 | LOOP1, CALL | Richiama la routine DELAY di ritardo di 10 ms |
| 014 | 277 | DELAY | Byte d'indirizzo LO di DELAY |
| 015 | 000 | – | Byte d'indirizzo HI di DELAY |
| 016 | 015 | DCR C | Decrementa il registro C |
| 017 | 302 | JNZ | Se il registro C non è uguale a 000, salta a LOOP1; altrimenti, passa all'istruzione successiva |
| 020 | 013 | LOOP1 | Byte d'indirizzo LO di LOOP1 |
| 021 | 003 | – | Byte d'indirizzo HI di LOOP1 |
| 022 | 005 | DCR B | Decrementa il registro B |
| 023 | 302 | JNZ | Se il registro B non è uguale a 000, salta a LOOP2; altrimenti, passa all'istruzione successiva |
| 024 | 005 | LOOP2 | Byte d'indirizzo LO di LOOP2 |
| 025 | 003 | – | Byte d'indirizzo HI di LOOP2 |
| 026 | 166 | HLT | Alt |

Occorreranno 1,0000345 secondi per acquisire ogni canale, o un totale di 64,0022 secondi per acquisire tutti e 64 i canali. Chiaramente, il tempo aggiuntivo richiesto per eseguire le istruzioni DCR, IN, CALL, INX, JNZ è trascurabile se confrontato con il ritardo di tempo di un secondo. Una subroutine a ritardo di tempo di 10 ms vi permetterà di acquisire dati ad una velocità fra 23,4 canali al minuto e 99,7 canali al secondo cambiando semplicemente il valore del byte di timing a LO = 012.

## TERZO PROGRAMMA: USCITA DA UN SISTEMA DI ACQUISIZIONE DATI

Supponiamo che abbiate memorizzato 64 canali in memoria di lettura/scrittura iniziando a HI = 003 e LO = 100 ed ora vogliate mettere in uscita ogni canale alla velocità di un canale al secondo su di un circuito a latch appropriato, come quello mostrato nelle Figure 20-6, 20-10, 20-12, 20-13. Che tipo di programma è necessario? La risposta a questa domanda è che occorre un programma quasi identico al Secondo Programma. Vanno modificati solo tre byte istruzioni dal Secondo Programma:

.
.

| | | | |
|---|---|---|---|
| 005 | 176 | MOV M,A | Sposta i contenuti della locazione di memoria M nell'accumulatore |
| 006 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore su un latch di uscita |
| 007 | 002 | 002 | Codice dispositivo per il latch di uscita |

.
.

Per il resto, il Secondo Programma può essere usato come scritto. Per esempio, nel Secondo Programma, avete già fornito byte istruzioni per (a) identificare la locazione di memoria M, (b) stabilire il numero di canali posti nella memoria di lettura/scrittura, (c) dare inizio ad un ritardo di tempo di un secondo fra ogni canale, e (d) arrestarsi dopo che tutti e 64 i canali sono stati messi in uscita. Potete eseguire ripetutamente il Secondo Programma modificato, che ora chiameremo Terzo Programma, iniziando a HI = 003 e LO = 000. Il Terzo Programma è ora un programma di uscita che non modifica i contenuti della memoria di lettura/scrittura.

## QUARTO PROGRAMMA: COME SI RIVELA UN CARATTERE ASCII

Restando valido il concetto di usare dispositivi di ingresso per inserire otto bit di dati, una volta che avete inserito i dati potete escogitare degli espedienti nella programmazione per sfruttare la potenza dell'8080A. Per esempio, supponiamo che il byte di dati in ingresso sia il codice ASCII a 8 bit da una tastiera standard ASCII che ha un'uscita TTL. Ogni volta che viene inserito un nuovo byte ASCII, viene provato per determinare se è o non è l'ASCII equivalente alla lettera "E", che ha un codice ASCII di 305. Se è una "E", il byte ASCII è messo in uscita ed anche immagazzinato nella memoria di lettura/scrittura. Se non lo è, il programma eseguirà immediatamente il loop di ritorno all'istruzione IN ed inserirà un nuovo byte ASCII. La fig. 22-2 mostra il semplice diagramma di flusso di questo programma.

Il programma è il seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START, IN | Inserisci il carattere ASCII |
| 001 | 004 | 004 | Codice dispositivo per la tastiera ASCII |

| | | | |
|---|---|---|---|
| 003 | 376 | CPI | Confronta i contenuti dell'accumulatore con il byte di dati seguente. Se i due byte sono identici, setta il flag di zero. Altrimenti, azzera (resetta) il flag di zero. |
| 004 | 305 | 305 | Codice ASCII per la lettera "E" |
| 005 | 302 | JNZ | Se il flag di zero è resettato, cioè a livello logico 0, salta a START; altrimenti, passa all'istruzione seguente |
| 006 | 000 | START | Byte d'indirizzo LO di START |
| 007 | 003 | – | Byte d'indirizzo HI di START |
| 010 | 323 | OUT | Poni in uscita il codice ASCII per la lettera "E" |
| 011 | 002 | 002 | Codice dispositivo per il latch di uscita |
| 012 | 062 | STA | Memorizza i contenuti dell'accumulatore nella locazione di memoria STORE |
| 013 | 200 | STORE | Byte d'indirizzo LO di STORE |
| 014 | 003 | – | Byte d'indirizzo HI di STORE |
| 015 | 166 | HLT | Alt |

L'istruzione immediata di confronto, CPI, a LO = 003 e LO = 004, vi permette di confrontare il byte ASCII 305 con i contenuti dell'accumulatore *senza alterare i contenuti dell'accumulatore.* Cambiano solo i flag. Se il byte ASCII per la lettera "E" e i contenuti dell'accumulatore sono identici, il flag di zero è settato ad 1; altrimenti, il flag di zero è resettato sul livello logico 0. La condizione del flag di zero è poi testata dall'istruzione JNZ per determinare se continuare o meno ad eseguire il loop.

Abbiamo osservato che l'istruzione di confronto è sofisticata e, all'occasione, difficile da usare propriamente. Come indicato nel Capitolo N. 18, i due flag che vengono esaminati dopo un'istruzione di confronto, come la CPI, sono il flag di zero e il flag di carry. Nel programma suddetto si possono inserire quattro diverse istruzioni di salto condizionato a LO = 005. Le domande relative a tali byte di istruzioni possono essere riassunte come segue:

| Byte istruzione a LO = 005 | Domande relative |
|---|---|
| 302 | Il carattere d'ingresso ASCII è la lettera "E"? Se non lo è, continua ad eseguire il loop finchè lo diventa. |
| 312 | Il carattere d'ingresso ASCII è un carattere qualunque invece della lettera "E"? Continua ad eseguire il loop finchè viene inserito un carattere ASCII invece di "E" |

322 Il carattere d'ingresso ASCII è "A", "B", "C", o "D"? Se non lo è, continua ad eseguire il loop finchè lo diventa.

332 Il carattere d'ingresso ASCII va da "E" a "Z"? Se non è così continua ad eseguire il loop finchè ciò si verifica.

Abbiamo esaminato il Quarto Programma usando ognuna delle quattro istruzioni di salto condizionato suddette. Con il codice ASCII equivalente alle lettere "D", "E", e "F", il programma lavora come previsto.

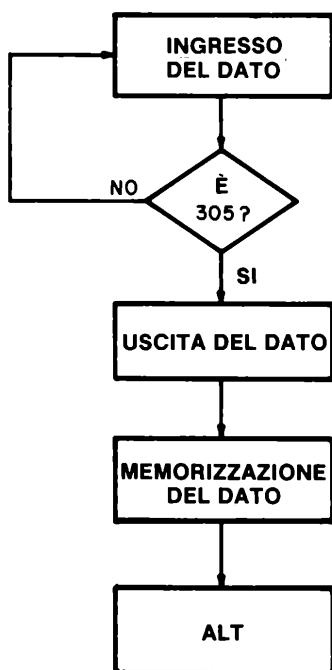



*Figura 22-2.* Diagramma di flusso del Quarto Programma, che esamina un carattere d'ingresso per determinare se è o meno il carattere ASCII "E". Quando viene inserito un "E", il programma mette in uscita il carattere, lo memorizza, e poi si arresta.

## ALTRI METODI PER GENERARE RITARDI DI TEMPO (DETTI ANCHE NUCLEI DI ATTESA)

Nel Secondo Programma di questo Capitolo, veniva usato un loop di ritardo di un secondo per rallentare la velocità alla quale il microcomputer acquisiva i dati da un dispositivo d'ingresso. L'uso di un tale loop rappresenta l'applicazione veramente inefficiente di un microcomputer, dal momento che il microcomputer potrebbe eseguire altre funzioni utili durante l'intervallo di un secondo. Altri metodi per generare ritardi di un secondo sono i seguenti:

- Un *clock a tempo reale* basato su una frequenza di 60 Hz.

  Viene prodotta un'onda quadra di 60 Hz da circuiti opportuni, ed usata per *interrompere* periodicamente l'esecuzione del programma. Come diremo nel Capitolo N. 23, il controllo di programma è diretto da una piccola subroutine che acquisisce il canale e poi rimanda il controllo al programma interrotto.

- Un *clock a tempo reale* basato su di un circuito oscillante con quarzo ad alta frequenza.

  Anche qui viene usato il sistema d'interruzione.

- Un *interval timer programmabile.*

  Un interval timer programmabile contiene (vedi il chip 8253) molti registri a 16 bit, i cui contenuti possono essere decrementati alla frequenza del microcomputer. Una parola di registro prosegue fino a che il contenuto del registro è zero, momento in cui viene mandato al microcomputer un impulso di interruzione. Un interval timer programmabile rappresenta un sistema di hardware-software combinati per risolvere il problema della generazione di ritardi di valore noto. Occorre un pò di hardware, cioè un circuito integrato, ma il valore del ritardo è stabilito con l'aiuto del software.

Se il microcomputer non ha altre funzioni da eseguire durante l'acquisizione, l'uso di un loop di attesa va benissimo.

## INTRODUZIONE AGLI ESPERIMENTI

I seguenti esperimenti forniscono esempi di circuiti di ingresso/uscita al microcomputer, con particolare riguardo all'acquisizione dati.

| Esperimento N. | Commento |
|---|---|
| 1 | Acquisizione veloce di dati. Dimostra un circuito di acquisizione dati e un programma che può acquisire dati da canali a 8 bit ad una velocità di 20.000 canali al secondo. |
| 2 | Acquisizione lenta di dati. L'aggiunta di una subroutine di ritardo rallenta la velocità alla quale i canali vengono acquisiti dal programma dell'Esperimento N. 1. |
| 3 | Come si rivela un carattere ASCII. Dimostra un programma che può rivelare l'ingresso di uno specifico carattere ASCII, come ASCII "E". |
| 4 | Collegamenti di un bus monitor (Vedi Esperimento N. 1 del Capitolo 17 per un circuito bus monitor che è basato sull'indicatore numerico TTL311 della Texas Instruments Incorporated). Descrive e dimostra il tipo di informazione sottoposta a latch, quando i segnali di controllo seguenti sono applicati all'ingresso di abilitazione del latch (STB) dell'indicatore numerico: $\overline{IN}$, $\overline{OUT}$, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{INTA}$, e impulsi di selezione dispositivo di ingresso e uscita. |
| 5 | Ingresso/uscita bidirezionale memory mapped usando un chip 8216. Dimostra l'uso del chip 8216 come una porta di I/O bidirezionale a 4 bit. |
| 6 | I/O tramite l'accumulatore, usando il chip 8255. L'interfaccia programmabile 8255 è largamente usata nei circuiti di ingresso/uscita. Questo esperimento dimostra l'operazione modo 0 di questo chip usando le tecniche di I/O tramite l'accumulatore. |
| 7 | I/O memory mapped usando il chip 8255. Cambiando gli ingressi dei segnali di controllo da $\overline{IN}$ e $\overline{OUT}$ in $\overline{MEMR}$ e $\overline{MEMW}$, è possibile convertire il circuito dell'Esperimento N.6 in operazione di I/O memory mapped. |
| 8 | Interfacciamento con un convertitore analogico-digitale. Viene illustrato un circuito di interfaccia tra un sistema microcomputer basato sul microprocessor 8080A ed il convertitore da digitale ad analogico AD7522 della Analog Devices. |
| 9 | Utilizzo di un convertitore a rampa. Con l'aiuto di uno specifico programma e di circuiteria addizionale basata sul comparatore LM311, diventa possibile convertire il circuito DAC dell'esperimento 8 in un convertitore da analogico a digitale. |

Alcuni degli esperimenti descritti esigono circuiti integrati costosi. Vi esortiamo a trattare tali c.i. con la dovuta cautela.

## ESPERIMENTO N. 1

## ACQUISIZIONE VELOCE DI DATI

### Scopo

Lo scopo di questo esperimento è far funzionare un semplice sistema di acquisizione dati basato sull'8080A che può acquisire dati ad alte velocità.

### Configurazioni dei pin dei circuiti integrati

7490

8095

### Schema del circuito

**Programma**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 100 | 100 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 003 | 003 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 006 | MVI B | Sposta il byte seguente nel registro B |
| 004 | 100 | 100 | Numero di canali che saranno acquisiti |
| 005 | 333 | LOOP, IN | Ingresso dei dati dal buffer |
| 006 | 004 | 004 | Codice dispositivo per il buffer d'ingresso. |
| 007 | 167 | MOV M,A | Sposta i contenuti dell'accumulatore nella locazione di memoria M indirizzata dai contenuti della coppia di registri H |
| 010 | 043 | INX H | Incrementa la coppia di registri H |
| 011 | 005 | DCR B | Decrementa il registro B |
| 012 | 302 | JNZ | Se il registro B non è uguale a 000, salta a LOOP ed acquisisci un altro canale dati; altrimenti, passa all'istruzione successiva |
| 013 | 005 | LOOP | Byte d'indirizzo LO di LOOP |
| 014 | 003 | – | Byte d'indirizzo HI di LOOP |
| 015 | 166 | HLT | Alt |

**Passo 1**

Studiate lo schema del circuito. Osservate che nel buffer d'ingresso vengono inseriti due contatori a decade in cascata 7490. Nel Capitolo N. 20, Figure 20-17 e 20-18 trovate due circuiti di buffer. Vi raccomandiamo di usare la coppia di chip 8095, dato che sono meno costosi e, secondo la nostra esperienza, meno soggetti ad essere danneggiati. Ottenete l'impulso di selezione dispositivo negativo da uno dei circuiti di decodifica descritti nel Capitolo N. 17.

Collegate il circuito richiesto per questo esperimento. La frequenza di clock dovrebbe essere inizialmente molto lenta, circa 1 Hz.

**Passo 2**

Caricate il programma nella memoria di lettura/scrittura da HI = 003 e LO = 000.

**Passo 3**

Eseguite il programma alla massima velocità.

Ora andate alla locazione di memoria HI = 003 e LO = 100 e, attraverso la memoria di lettura/scrittura, passate fino a HI = 003 e LO = 200. Che cosa osservate? Perchè?

Abbiamo osservato una lettura di 120, in codice ottale, per tutte le locazioni di memoria che iniziano da LO = 100 e finiscono a LO = 177. Tale lettura corrisponde ad una parola binaria a 8 bit di 01010000, che è equivalente al numero decimale 50 in BCD impaccato. Il microcomputer ha eseguito il programma così velocemente che è stata inserita una sola uscita della coppia di contatori 7490 in tutte le locazioni di memoria.

**Passo 4**

Che cosa intendiamo per "BCD impaccato"?

BCD (decimale codificato binario) è un codice binario a quattro bit per le cifre decimali da 0 a 9. Due cifre BCD comprendono un totale di otto bit, che possono essere inseriti sia in microcomputer a 8 bit sia nell'8080A. Per "BCD impaccato" intendiamo un canale dati a otto bit che contiene due cifre BCD a quattro bit.

**Passo 5**

Eseguite il programma alla massima velocità per parecchie volte. Osservate quali dati inserire iniziando alla locazione di memoria HI = 003 e LO = 100. Che cosa concludete?

In ogni caso, inseriamo solo una coppia di cifre BCD nella memoria di lettura/scrittura.

**Passo 6**

Abbiamo precedentemente concluso, in questo Capitolo, che occorrevano 37 cicli per acquisire un solo canale dati a 8 bit. Se un microcomputer opera ad una cadenza di temporizzazione di 750 kHz, quanto tempo occorre per acquisire un solo canale? Qual'è la velocità di acquisizione dati, in byte al secondo?

A 750 kHz, un ciclo dura 1,333 microsecondi. Perciò, 37 cicli corrispondono a 49,3 microsecondi e ad una velocità di 20,27 kHz.

**Passo 7**

Se l'ingresso di clock alla coppia di contatori 7490 ha una frequenza di 20 kHz, che cosa potete concludere circa i dati memorizzati nella memoria di lettura/scrittura, iniziando da HI = 003 e LO = 100?

Ci aspetteremmo di vedere una serie di conteggi sempre maggiori iniziando da HI = 003 e LO = 100, con un incremento di un conteggio fra una locazione di memoria e la successiva. La ragione è che la frequenza clock d'ingresso ai chip 7490 è identica alla velocità di acquisizione dei dati del microcomputer a 750 kHz. Per esempio, quando abbiamo eseguito un esperimento di questo tipo sul nostro microcomputer, abbiamo osservato il seguente risultato:

| Byte d'indirizzo LO | Dati memorizzati | |
|---|---|---|
| della memoria di lettura/scrittura | Codice ottale | BCD impaccati |
| 100 | 106 | 46 |
| 101 | 107 | 47 |
| 102 | 110 | 48 |
| 103 | 111 | 49 |
| 104 | 120 | 50 |
| 105 | 121 | 51 |
| 106 | 122 | 52 |
| 107 | 123 | 53 |
| 110 | 124 | 54 |
| 111 | 125 | 55 |
| 112 | 126 | 56 |

**Passo 8**

Posizionate la frequenza di clock dell'ingresso clock dei contatori 7490 su 20 kHz o poco meno. Le frequenze che vanno da 5 a 20 kHz andrebbero benissimo. Eseguite una volta il programma. Osservate i dati memorizzati iniziando da HI = 003 e LO = 100. Che cosa concludete?

I valori dal contatore 7490 sono memorizzati sequenzialmente in 64 locazioni di memoria di lettura/scrittura iniziando da HI = 003 e LO = 100. Il nostro primo canale dati era l'ottale 070 e l'ultimo ottale 166.

**Passo 9**

Se occorrono 49,33 microsecondi per acquisire un solo byte di dati, e se il primo byte di dati è 070 e l'ultimo è 166, qual'è la frequenza del contatore?

Il tempo totale richiesto per acquisire 64 byte di dati è 49,33 microsecondi per 64 canali = 3157,12 microsecondi. Il numero di conteggi fra il primo e l'ultimo byte di dati è,

Dati del primo conteggio = $070_8 = 01111000_2 = 38_{16} = 56_{10}$

Dati dell'ultimo conteggio = $166_8 = 01110110_2 = 76_{16} = 118_{10}$

In altre parole, in 3,15712 ms sono stati acquisiti in tutto 118 − 56 = 62 byte di dati. La frequenza di clock d'ingresso è quindi,

Frequenza clock = 62/0,00315712 secondi

= 19,64 kHz

**Passo 10**

Calcolate le frequenze di clock d'ingresso del vostro circuito usando la procedura di calcolo descritta in precedenza.

| Conteggio iniziale (LO=100) | Conteggio finale (LO=177) | Frequenza calcolata kHz |
|---|---|---|

Abbiamo ottenuto i seguenti risultati, dove i conteggi sono dati in decimale:

| Conteggio iniziale (decimale) | Conteggio finale (decimale) | Frequenza calcolata kHz |
|---|---|---|
| 09 | 305 | 93,8 |
| 20 | 147 | 40,2 |
| 11 | 105 | 30,0 |
| 36 | 74 | 12,0 |

## ESPERIMENTO N. 2

## ACQUISIZIONE LENTA DI DATI

### Scopo

Lo scopo di questo esperimento è operare con un semplice sistema di acquisizione dati basato sull'8080A che può acquisire dati a bassa velocità.

### Schema del circuito

Vedere l'esperimento precedente per i particolari del circuito del contatore 7490.

### Programma

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 100 | 100 | Byte del registro L, il byte d'indirizzo HI della locazione di memoria M |
| 002 | 003 | 003 | Byte del registro H, il byte d'indirizzo della locazione di memoria M |
| 003 | 006 | MVI B | Carica il registro B con il byte seguente |
| 004 | 100 | 100 | Numero di canali che saranno acquisiti |
| 005 | 333 | LOOP, IN | Ingresso dati dalla coppia di contatori |
| 006 | 004 | 004 | Codice dispositivo per il buffer del contatore |
| 007 | 167 | MOV M,A | Sposta il contenuto dell'accumulatore nella locazione di memoria M, indirizzata dai contenuti della coppia di registri H |
| 010 | 043 | INX H | Incrementa la coppia di registri H |
| 011 | 016 | MVI C | Carica il registro C con il byte di timing seguente |
| 012 | 144 | 144 | Byte di timing per il registro C |
| 013 | 315 | LOOP1, CALL | Richiama la routine DELAY di ritardo di 10 ms |
| 014 | 277 | DELAY | Byte d'indirizzo LO di DELAY |
| 015 | 000 | – | Byte d'indirizzo HI di DELAY |
| 016 | 015 | DCR C | Decrementa il registro C |
| 017 | 302 | JNZ | Se il registro C non è uguale a 000, salta a LOOP1; altrimenti, passa all'istruzione successiva |

| | | | |
|---|---|---|---|
| 020 | 013 | LOOP1 | Byte d'indirizzo LO di LOOP1 |
| 021 | 003 | – | Byte d'indirizzo HI di LOOP1 |
| 022 | 005 | DCR B | Decrementa il registro B |
| 023 | 302 | JNZ | Se il registro B non è uguale a 000, salta a LOOP; altrimenti, passa all'istruzione successiva |
| 024 | 005 | LOOP | Byte d'indirizzo LO di LOOP |
| 026 | 003 | – | Byte d'indirizzo HI di LOOP |
| 026 | 166 | HLT | Alt |

**Passo 1**

Il circuito è identico a quello usato nell'esperimento precedente. Caricare il nuovo programma nella memoria di lettura/scrittura partendo da HI = 003 e LO = 000.

**Passo 2**

Eseguite il programma. Ricordate che non sarà necessario un secondo per ogni canale, o un totale di 64 secondi prima che tutti i canali dati siano acquisiti e il programma si arresti. Il vostro ingresso di clock ai contatori 7490 dovrebbe essere di circa 1 Hz.

Andate alla locazione di memoria HI = 003 e LO = 100 e passate attraverso la memoria di lettura/scrittura. Che cosa osservate?

Abbiamo osservato i dati che abbiamo acquisito ad un rate di un canale al secondo.

**Passo 3**

Ora apportate i seguenti cambiamenti al programma e aggiungete una porta di uscita di indirizzo 002, se non ce n'è una ancora disponibile sul vostro microcomputer.

| | | | |
|---|---|---|---|
| 005 | 176 | MOV M,A | Sposta i contenuti della locazione di memoria M nell'accumulatore |
| 006 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore su un latch di uscita |
| 007 | 002 | 002 | Codice dispositivo del latch di uscita |

Eseguite il programma e spiegate che cosa osservate sul latch, che dovrebbe essere collegato o agli otto indicatori a LED o ad una coppia di display a sette segmenti.

Abbiamo osservato gli stessi ingressi dati che abbiamo memorizzato nella memoria di lettura/scrittura prima di modificare il programma! In altre parole, modificando tre byte d'istruzioni, siamo stati in grado di convertire il nostro programma di ingresso dati in un programma di uscita dei dati. Ogni canale dati è stato messo in uscita ad una velocità di un canale al secondo, così è stato molto semplice studiare i dati che avevamo inizialmente immagazzinato in memoria.

**Passo 4**

Ripetete i passi 2 e 3 tutte le volte che volete. Ogni volta che volete inserire i dati nella memoria dovete assicurarvi che nelle locazioni di memoria da LO = 005 e LO = 007 siano presenti i byte istruzioni esatti.

**Passo 5**

Invece di modificare il programma ogni volta, probabilmente sarebbe più conveniente caricare un programma di uscita separato partendo da HI = 003 e LO = 030. Supponendo che facciate in questo modo, quali sarebbero gli indirizzi di LOOP e LOOP1?

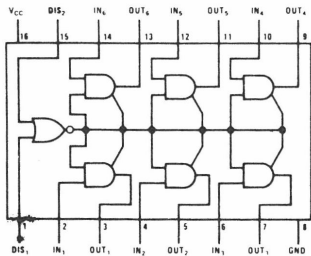
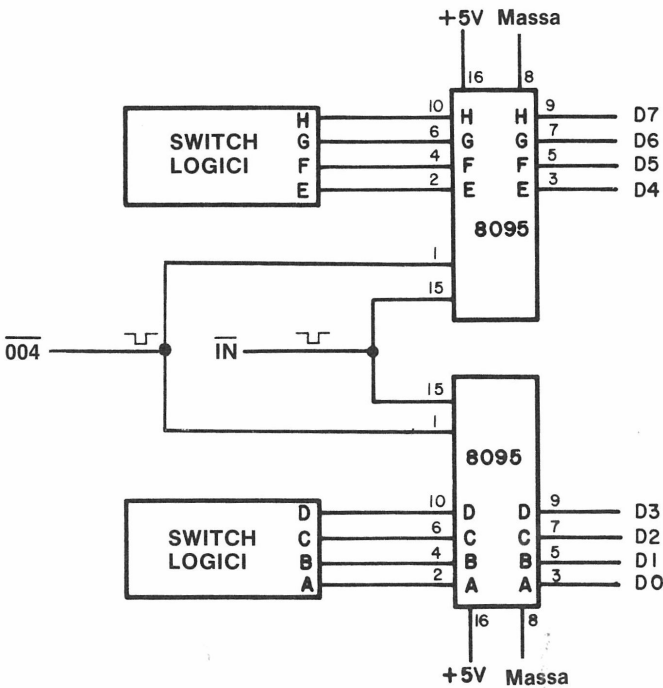
LOOP sarebbe a HI = 003 e LO = 035 e LOOP1 sarebbe da HI = 003 e LO = 043.

## ESPERIMENTO N. 3

## COME SI RIVELA UN CARATTERE ASCII

### Scopo

Lo scopo di questo esperimento è cablare un'interfaccia ed eseguire un programma che dimostri come rivelare il carattere ASCII "E".

### Configurazione dei pin del circuito integrato

### Schema del circuito

**Programma**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| START: 000 | 333 | IN | Ingresso del carattere ASCII dallo switch |
| 001 | 004 | 004 | Codice dispositivo per la porta d'ingresso |
| 002 | 376 | CPI | Confronta i contenuti dell'accumulatore con il byte di dati seguente. Se i due byte sono identici, setta il flag di zero. Se no, resetta il flag di zero. Setta il flag di carry se 305 è maggiore dei contenuti dell'accumulatore. |
| 003 | 305 | 305 | Codice ASCII per la lettera "E" |
| 004 | 302 | JNZ | Se il flag di zero è resettato, salta indietro a START; altrimenti, passa all'istruzione successiva. |
| 005 | 000 | – | Byte d'indirizzo LO di START |
| 006 | 003 | – | Byte d'indirizzo HI di START |
| 007 | 323 | OUT | Poni in uscita il codice ASCII per la lettera "E", che è contenuta nell'accumulatore |
| 010 | 002 | 002 | Codice dispositivo per la porta di uscita |
| 011 | 166 | HLT | Alt |

**Passo 1**

Cablate il circuito di interfaccia nello schema. Caricate il programma nella memoria di lettura/scrittura partendo da HI = 003 e LO = 000.

**Passo 2**

Settate gli switch logici dei buffer three-state 8095 sul valore ASCII equivalente alla lettera "D", cioè 304 in codice ottale o 11000100 in binario. Eseguite il programma alla massima velocità. Che cosa succede?

Nel nostro caso, non succede niente. La porta di uscita 002 non ha rivelato 305.

**Passo 3**

Posizionate gli switch logici su 306 mentre il microcomputer opera. Assicuratevi di non averlo posizionato su 305, anche momentaneamente! Nessun cambiamento?

No. La ragione è che fino ad ora non è stato rivelato un 305. Stando così le cose, il programma continua ad eseguire LOOP tornando a START.

**Passo 4**

Ora cambiate gli switch logici in 305. Che cosa succede?

Il codice ASCII per "E", 305, è messo in uscita sulla porta 002 e il microcomputer si arresta. Fa così, in risposta al quesito posto dal byte istruzioni LO = 004:

| | | | |
|---|---|---|---|
| 004 | 302 | JNZ | Il carattere ASCII è la lettera "E"? Se no, continua ad eseguire loop tornando indietro a START, finchè lo diventa. |

**Passo 5**
Cambiate il byte istruzioni a LO = 004 in

| | | | |
|---|---|---|---|
| 005 | 312 | JZ | Il carattere in ingresso ASCII è un qualunque carattere diverso dalla lettera "E"? Continua ad eseguire il loop finchè viene inserito un carattere diverso da "E". |

Posizionate gli switch logici su 305 ed esguite alla massima velocità. Che cosa osservate?

Il programma continua ad eseguire il loop. Durante ogni loop, rivela il carattere "E".

**Passo 6**

Ora posizionate gli switch logici su 304 ed eseguite ancora una volta il programma. Che cosa succede?

Il microcomputer si arresta immediatamente.

**Passo 7**

Potete anche sostituire uno dei seguenti byte istruzioni a LO = 004.

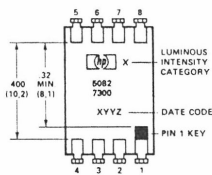
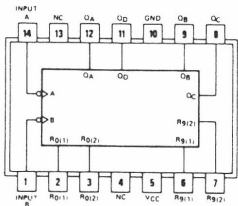
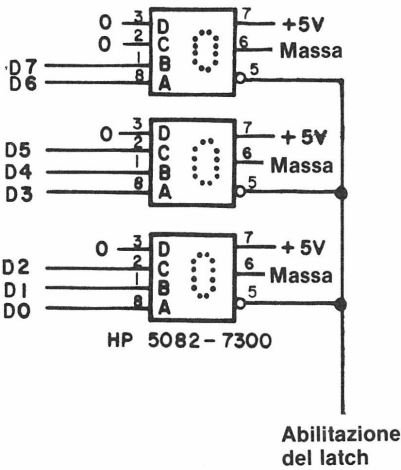
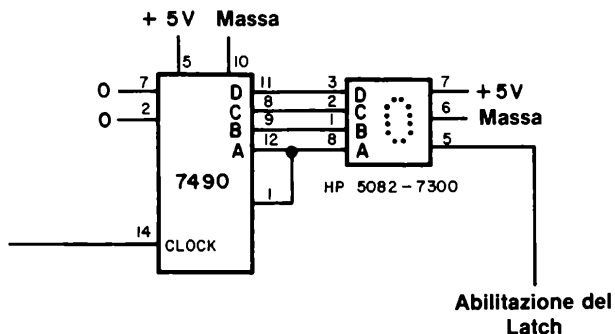
| | | | |
|---|---|---|---|
| 004 | 322 | JNC | Il carattere d'ingresso ASCII è minore di 305? Se no, continua ad eseguire il loop finchè lo diventa. |
| | oppure | | |
| 004 | 332 | JC | Il carattere in ingresso ASCII è maggiore di 304? Se no, continua ad eseguire il loop finchè lo diventa. |

## ESPERIMENTO N. 4

## DEFINIZIONE DI UN BUS MONITOR

### Scopo

Lo scopo di questo esperimento è montare una coppia di circuiti che possono essere utili nei seguenti esperimenti: (1) un *bus monitor* a tre cifre ottali, che vi permette di visualizzare tutte le informazioni che passano sul bus di dati bidirezionali, e (2) un contatore a latch 7490, che vi permette di rivelare e conteggiare diversi tipi di impulsi di sincronizzazione.

### Configurazione dei pin dei circuiti integrati

| PIN | FUNCTION 5082-7300 and 7302 Numeric | FUNCTION 5082-7340 Hexadecimal |
|---|---|---|
| 1 | Input 2 | Input 2 |
| 2 | Input 4 | Input 4 |
| 3 | Input 8 | Input 8 |
| 4 | Decimal point | Blanking control |
| 5 | Latch enable | Latch enable |
| 6 | Ground | Ground |
| 7 | $V_{cc}$ | $V_{cc}$ |
| 8 | Input 1 | Input 1 |

### Schema del circuito

**Passo 1**

Montate i circuiti mostrati nelle figure, preferibilmente su di una sola piastra SK-10. Li troverete utili come monitor degli impulsi di selezione dispositivo, dei segnali di controllo, e dei dati che appaiono sul bus di dati bidirezionali.

**Passo 2**

Il display 5082-7300 della Hewlett Packard contiene un latch a quattro bit del tipo 7475, che è abilitato da un impulso STROBE di livello logico 0. Che cosa significa questo?

Il latch 7475 è un latch di tipo D che segue l'ingresso quando il latch è abilitato. Perciò, un livello logico 0 applicato all'HP 5082-7300 significa che all'uscita sarà la stessa dell'ingresso fino a che l'impulso di STROBE rimane allo stato logico 0. Sul fronte positivo dell'impulso d'ingresso di STROBE viene eseguito il LATCH dei dati in ingresso.

**Passo 3**

Si possono usare molti segnali di controllo ed altri segnali come ingresso di STROBE ai latch/display 5082-7300 che sono collegati al bus di dati bidirezionale da D0 a D7. Fate un elenco di alcuni di questi segnali e spiegate quali informazioni vi permettono di sottoporre a latch dal bus di dati.

Ecco alcuni segnali utili:

| | |
|---|---|
| $\overline{\text{OUT}}$ | Esegue un latch di tutti i dati in uscita per mezzo dell'istruzione OUT |
| $\overline{\text{IN}}$ | Esegue un latch di tutti i dati in ingresso per mezzo dell'istruzione IN |
| $\overline{\text{MEMR}}$ | Esegue un latch di tutti i dati in ingresso per mezzo di un'istruzione del tipo lettura in memoria |
| $\overline{\text{MEMW}}$ | Esegue un latch di tutti i dati in uscita per mezzo di un'istruzione del tipo scrittura in memoria |
| Impulso di uscita DS | Esegue un latch di tutti i dati in uscita verso uno specifico dispositivo di uscita |
| Impulso d'ingresso DS | Esegue un latch di tutti i dati in ingresso provenienti da uno specifico dispositivo d'ingresso |
| $\overline{\text{INTA}}$ | Esegue un latch dei dati che appaiono sul bus durante un segnale di controllo di riconoscimento di un'interruzione. |

Chiameremo il circuito a tre cifre che esegue un latch sul bus di dati bidirezionale, da D0 a D7, *bus monitor*, dato che esso permette di visualizzare tutte le informazioni che appaiono sul bus di dati. Userete questo monitor negli esperimenti successivi, perciò conservatelo.

**Passo 4**

Il secondo circuito di questo esperimento consiste in un latch/display della Hewlett Packard posto all'uscita di un contatore 7490. Qual'è la funzione di questo circuito?

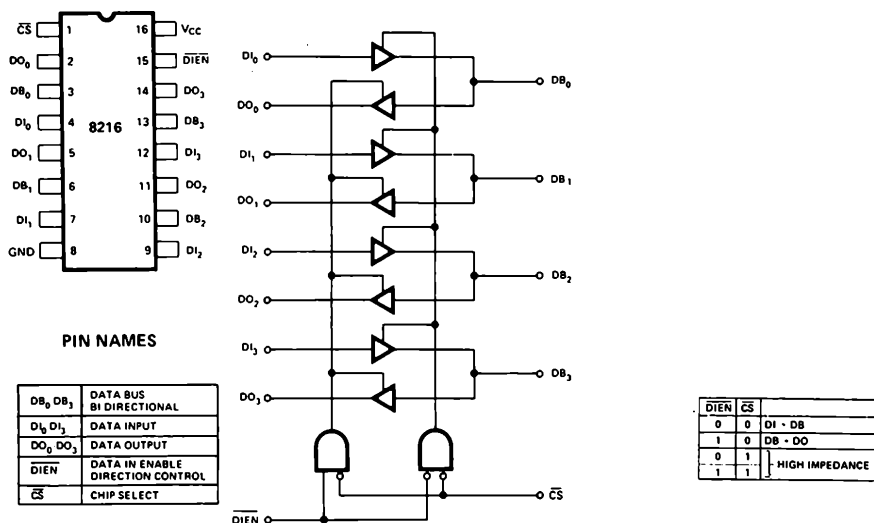
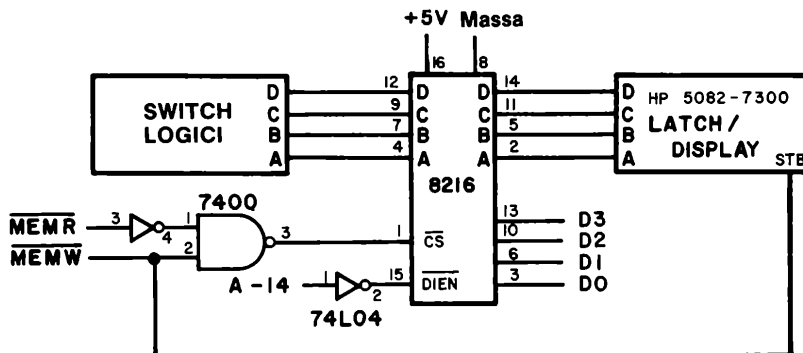
Il circuito vi permette di rivelare il segnale di controllo individuale o gli impulsi di selezione dispositivo, ammesso che ne vengano generati solo pochi. Un circuito di questo tipo è generalmente usato quando il programma contiene un'istruzione di Alt o quando vi sono intervalli di tempo lunghi fra un'impulso e l'altro.

**ESPERIMENTO N. 5**

## I/O MEMORY MAPPED BIDIREZIONALE REALIZZATO UTILIZZANDO UN CHIP 8216

### Scopo

Lo scopo di questo esperimento è operare su un chip 8216 e su una porta di I/O bidirezionale utilizzata secondo la tecnica memory mapped.

### Configurazione dei pin e schema logico del circuito integrato

| | |
|---|---|
| DB0 DB3 | DATA BUS BI DIRECTIONAL |
| DI0 DI3 | DATA INPUT |
| DO0 DO3 | DATA OUTPUT |
| DIEN | DATA IN ENABLE DIRECTION CONTROL |
| CS | CHIP SELECT |

| DIEN | CS | |
|---|---|---|
| 0 | 0 | DI → DB |
| 1 | 0 | DB → DO |
| 0 | 1 | HIGH IMPEDANCE |
| 1 | 1 | HIGH IMPEDANCE |

### Schema del circuito

**Programma**

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | START, LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 000 | 000 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 100 | 100 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 176 | MOV A,M | Sposta i contenuti della locazione di memoria M nell'accumulatore |
| 004 | 167 | MOV M,A | Sposta i contenuti dell'accumulatore nella locazione di memoria M |
| 005 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore sulla porta di uscita 002 |
| 006 | 002 | 002 | Codice dispositivo per la porta 002 |
| 007 | 303 | JMP | Salto incondizionato alla locazione di memoria START |
| 010 | 000 | START | Byte d'indirizzo LO di START |
| 011 | 003 | – | Byte d'indirizzo HI di START |

**Passo 1**

Montate il circuito mostrato. Caricate il programma nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 000.

**Passo 2**

Eseguite il programma. Che cosa osservate sulla porta 002 e anche sul latch/display quando cambiate gli switch logici da 0000 a 1001?

I quattro bit meno significativi nella porta di uscita 002 e nel latch/buffer mostrano la stessa lettura dell'ingresso dello switch logico all'8216. I quattro bit più significativi nella porta di uscita 002 rimangono tutti a livello logico 1. Il programma legge i dati dallo switch logico usando l'istruzione MOV A,M, e poi pone in uscita i contenuti dell'accumulatore sul latch/display 8216 usando l'istruzione MOV M,A. Infine, l'istruzione OUT 002 mette in uscita i contenuti dell'accumulatore sulla porta di uscita.

Nelle seguenti tabelle della verità, aggiungete i dati relativi agli ingressi $\overline{DIEN}$ e $\overline{CS}$ del chip 8216.

| A-14 | $\overline{CS}$ |
|---|---|
| 0 | |
| 1 | |

| $\overline{MEMR}$ | $\overline{MEMW}$ | $\overline{DIEN}$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

| DIEN | CS | Operazione del chip 8216 |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

Spiegate il significato di queste tabelle nello spazio seguente.

La prima tabella della verità indica se il chip è abilitato o no:

| A-14 | CS | Operazione del chip 8216 |
|---|---|---|
| 1 | 0 | Chip abilitato |
| 0 | 1 | Chip disabilitato (stato di alta impedenza) |

La seconda tabella della verità indica la direzione del trasferimento dei dati attraverso il chip:

| $\overline{MEMR}$ | $\overline{MEMW}$ | $\overline{DIEN}$ | |
|---|---|---|---|
| 0 | 0 | 0 | Non permesso |
| 0 | 1 | 0 | Lettura in memoria |
| 1 | 0 | 1 | Scrittura in memoria |
| 1 | 1 | 1 | Scrittura in memoria |

La tabella della verità finale riassume l'operazione del chip 8216

| DIEN | CS | Operazione del chip 8216 |
|---|---|---|
| 0 | 0 | Ingresso dei dati nel chip 8080A |
| 0 | 1 | Chip disabilitato |
| 1 | 0 | Uscita dei dati dal chip 8080A |
| 1 | 1 | Chip disabilitato |

In altre parole, quando $\overline{MEMR}$ e $\overline{CS}$ sono entrambi a livello logico 0, l'8216 serve come porta d'ingresso. Quando $\overline{MEMW}$ e $\overline{CS}$ sono entrambi a livello logico 0, l'8216 serve come porta di uscita. $\overline{MEMR}$ e $\overline{MEMW}$ non possono essere entrambi a livello logico 0 dato che il microcomputer 8080A non può leggere e scrivere nello stesso tempo. Lo stato $\overline{MEMW}$ = $\overline{MEMR}$ = 0 non viene mai osservato dalla logica di controllo esterna.

**Passo 3**

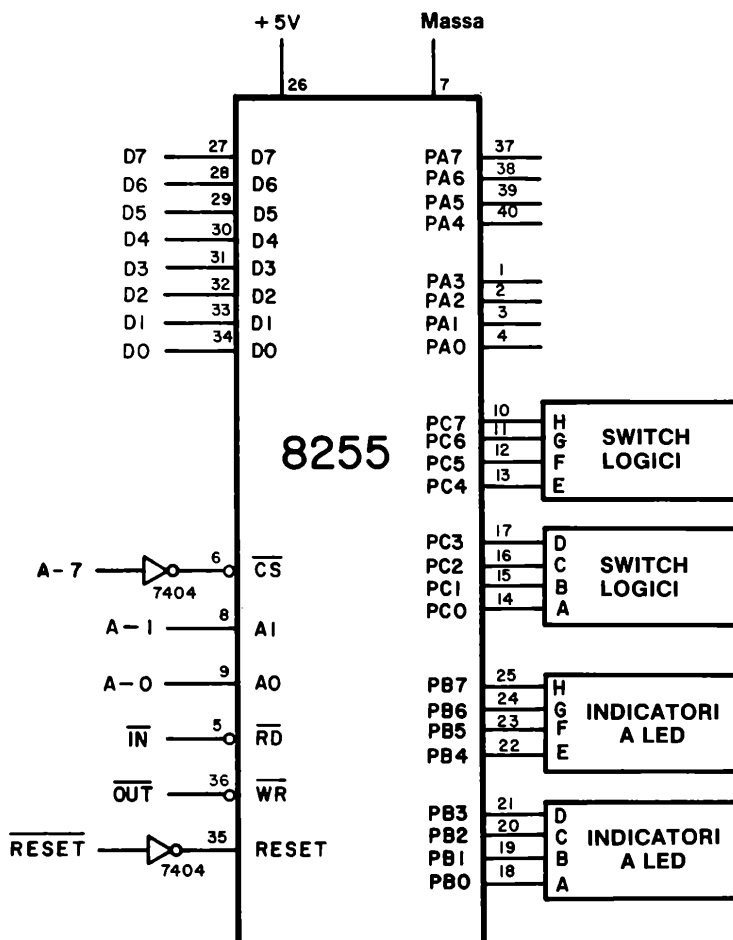
Questo chip è utile come porta di I/O?

Forse, ma è necessario un latch aggiuntivo se va usato come porta di uscita. In genere, l'8216 è usato come driver/buffer del bus bidirezionale, che ha un fan-in di 0,1 e un fan-out di 30.

**ESPERIMENTO N. 6**

## I/O TRAMITE L'ACCUMULATORE REALIZZATO CON L'USO DEL CHIP 8255

### Scopo

Lo scopo di questo esperimento è di mostrare l'uso dell'interfaccia periferica programmabile 8255 come porta di I/O con la tecnica utilizzante l'accumulatore.

### Schema del circuito

**Programma**

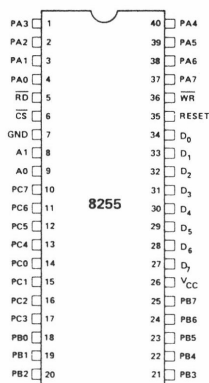
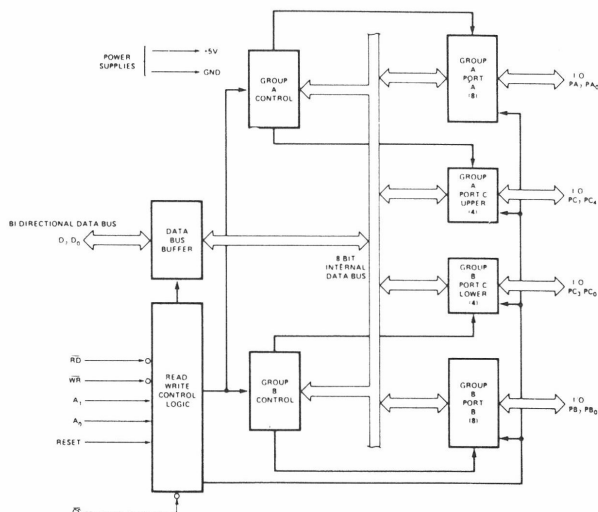
| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 076 | MVI A | Sposta la parola di controllo seguente nell'accumulatore |
| 001 | 231 | 231 | Parola di controllo che stabilisce l'operazione di modo 0 del chip 8255, con le porte A e C porte d'ingresso e la porta B porta di uscita |
| 002 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore sul latch di uscita seguente |
| 003 | 203 | 203 | Codice dispositivo per il registro di controllo all'interno del chip 8255 |
| 004 | 333 | LOOP, IN | Dati in ingresso dallo switch logico alla porta C |
| 005 | 202 | 202 | Codice dispositivo per la porta C |
| 006 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore sulla porta B |
| 007 | 201 | 201 | Codice dispositivo per la porta B |
| 010 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 011 | 004 | LOOP | Byte d'indirizzo LO di LOOP |
| 012 | 003 | – | Byte d'indirizzo HI di LOOP |

**Configurazione dei pin e schema blocchi del circuito integrato**

La tabella della verità per le tre porte di I/O e per il registro di controllo è la seguente:

8255 BASIC OPERATION

| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | INPUT OPERATION (READ) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | PORT A → DATA BUS |
| 0 | 1 | 0 | 1 | 0 | PORT B → DATA BUS |
| 1 | 0 | 0 | 1 | 0 | PORT C → DATA BUS |
| | | | | | **OUTPUT OPERATION (WRITE)** |
| 0 | 0 | 1 | 0 | 0 | DATA BUS → PORT A |
| 0 | 1 | 1 | 0 | 0 | DATA BUS → PORT B |
| 1 | 0 | 1 | 0 | 0 | DATA BUS → PORT C |
| 1 | 1 | 1 | 0 | 0 | DATA BUS → CONTROL |
| | | | | | **DISABLE FUNCTION** |
| X | X | X | X | 1 | DATA BUS → 3-STATE |
| 1 | 1 | 0 | 1 | 0 | ILLEGAL CONDITION |

La configurazione dei pin e lo schema a blocchi del chip 8255 sono:

| | |
|---|---|
| $D_7$–$D_0$ | DATA BUS (BI-DIRECTIONAL) |
| RESET | RESET INPUT |
| $\overline{CS}$ | CHIP SELECT |
| $\overline{RD}$ | READ INPUT |
| $\overline{WR}$ | WRITE INPUT |
| A0, A1 | PORT ADDRESS |
| PA7-PA0 | PORT A (BIT) |
| PB7-PB0 | PORT B (BIT) |
| PC7-PC0 | PORT C (BIT) |
| $V_{CC}$ | +5 VOLTS |
| GND | Ø VOLTS |

**Passo 1**

Studiate la tabella della verità delle tre porte di I/O e del registro di controllo all'interno del chip 8255. I bit del bus d'indirizzo A-0, A-1, e A-7 sono all'interno per selezionare la porta specifica o il registro desiderato. I segnali di controllo $\overline{IN}$ e $\overline{OUT}$ sono collegati rispettivamente a $\overline{RD}$ e $\overline{WR}$. Perciò, il codice dispositivo a otto bit, da A0 ad A7, identifica il dispositivo di I/O particolare associato all'istruzione IN o OUT. Dato che il bus d'indirizzo non è decodificato in assoluto, i bit d'indirizzo da A2 ad A6 possono essere 0 o 1. Nel definire i codici dispositivo, abbiamo lasciato questi bit a livello logico 0.

| Codice dispositivo | Porta di I/O o registro |
|---|---|
| 200 | Porta A |
| 201 | Porta B |
| 202 | Porta C |
| 203 | Registro di controllo |

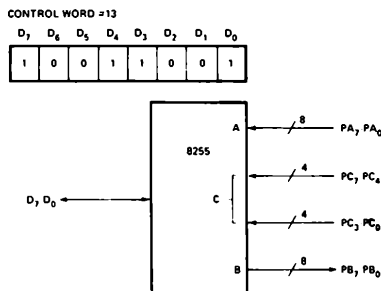
Nel programma, i byte istruzioni a LO = 003, 005 e 007 sono tutti codici dispositivo.

**Passo 2**

Montate il circuito mostrato. Usate il bit A-7 del bus d'indirizzo, per l'ingresso $\overline{CS}$, $\overline{IN}$ per l'ingresso $\overline{RD}$, e $\overline{OUT}$ per l'ingresso $\overline{WR}$. Questo uso del chip è un esempio dell'I/O tramite l'accumulatore.

**Passo 3**

Caricate il programma nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 000. Quale pensate che sia il significato del byte di istruzioni a LO = 001?

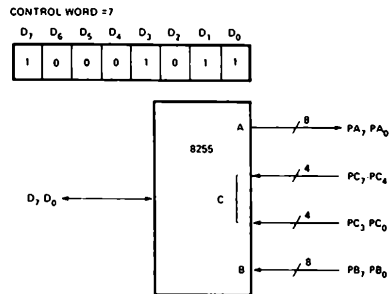
Come stabilito nel programma, è una parola di controllo che determina l'operazione modo 0 del chip 8255 e se le porte A, B e C sono porte d'ingresso o di uscita. In questo caso, la parola di controllo che corrisponde alle porte A e C, che sono porte d'ingresso, e alla porta B, che è una porta di uscita, è:

**Passo 4**

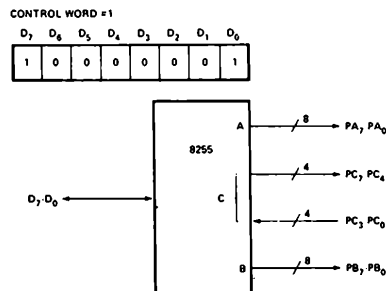
Eseguite il programma. Mentre il microcomputer opera, cambiate la disposizione dello switch logico ed osservate l'uscita alla porta B. Che cosa succede?

La porta B visualizza l'ingresso dello switch logico alla porta C. Qualunque cambiamento nella disposizione dello switch logico avviene istantaneamente alla porta B. Questo comportamento dimostra che abbiamo inserito dei dati nell'accumulatore e che li abbiamo messi in uscita dall'accumulatore sulla porta B.

**Passo 5**

Cambiate la parola di controllo a LO = 001 in 213. Eseguite il programma ed osservate se c'è o no un'uscita alla porta B quando cambiate la disposizione dello switch logico alla porta C.

Abbiamo osservato che non vi sono cambiamenti nell'uscita alla porta B, che rimane a 000. La ragione è che la parola di controllo impone ora che la porta B sia una porta d'ingresso,

**Passo 6**

Infine, cambiate la parola di controllo a LO = 001 in 201 che corrisponde alla situazione seguente,

Notate che ora, solo quattro dei bit nella porta C sono bit d'ingresso. Rimuovete gli switch logici dai bit da PC4 a PC7. Cambiate la parola di controllo a LO = 001 ed eseguite il programma. Che cosa osservate?

L'uscita alla porta B rispecchia l'ingresso alla porta C dei bit da PC0 a PC3. La porta B è di nuovo una porta di uscita.

**Passo 7**

Il chip 8255 è un'interfaccia interessante ma alquanto complicata, costruita dalla Intel Corporation. Per ulteriori dettagli, procuratevi una copia del "8080 Microcomputer Peripherals User's Manual" pubblicato di recente ed una copia del "8255 Programmable Peripheral Interface Applications".

*Conservate questo circuito per il prossimo esperimento, nel quale userete il chip 8255 come dispositivo di I/O memory mapped.*

## ESPERIMENTO N. 7

## I/O MEMORY MAPPED CON L'USO DEL CHIP 8255

### Scopo

Lo scopo di questo esperimento è di mostrare l'uso dell'interfaccia programmabile 8255 come porta di I/O memory mapped. La configurazione dei pin del circuito integrato e lo schema del circuito sono stati dati nell'esperimento precedente, nel quale vi è stato chiesto di conservare il circuito.

### Programma

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 041 | LXI H | Carica la coppia di registri H con i due byte seguenti |
| 001 | 003 | 003 | Byte del registro L, il byte d'indirizzo LO della locazione di memoria M |
| 002 | 200 | 200 | Byte del registro H, il byte d'indirizzo HI della locazione di memoria M |
| 003 | 006 | MVI B | Sposta la parola di controllo seguente nel registro B |
| 004 | 231 | 231 | Parola di controllo che stabilisce l'operazione modo 0 del chip 8255, con le porte A e C che sono porte d'ingresso e la porta B che è una porta di uscita |
| 005 | 160 | MOV M,B | Sposta i contenuti del registro B nella locazione di memoria M, che è il registro di controllo all'interno del chip |
| 006 | 055 | DCR L | Decrementa il registro L |
| 007 | 176 | LOOP, MOV A,M | Ingresso dei dati dallo switch logico, attraverso la porta C |
| 010 | 055 | DCR L | Decrementa il registro L |
| 011 | 167 | MOV M,A | Poni in uscita i contenuti dell'accumulatore sulla porta B |
| 012 | 054 | INR L | Incrementa il registro L |
| 013 | 303 | JMP | Salto incondizionato alla locazione di memoria LOOP |
| 014 | 007 | LOOP | Byte d'indirizzo LO di LOOP |
| 015 | 003 | – | Byte d'indirizzo HI di LOOP |

**Passo 1**

L'ingresso di $\overline{CS}$ dovrebbe ora essere il bit invertito A-15. Gli ingressi di $\overline{RD}$ e $\overline{WR}$ dovrebbero essere ora $\overline{MEMR}$ e $\overline{MEMW}$, rispettivamente. Apportate queste modifiche ai collegamenti del circuito mostrato nell'esperimento precedente.

**Passo 2**

Caricate il programma nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 000.

**Passo 3**

Eseguite il programma. Variate la disposizione dello switch logico ed osservate che cosa accade alla porta B. Che cosa potete concludere?

L'ingresso dello switch logico alla porta C, appare come l'uscita di un indicatore a LED alla porta B.

**Passo 4**

Cambiate la parola di controllo a LO = 004 in 213 e poi in 201. Eseguite il programma in entrambi i casi. Spiegate che cosa osservate alla porta B.

Con la parola di controllo 213, la porta B non funziona più come una porta di uscita. Per la parola di controllo 201, la porta B è di nuovo una porta di uscita, ma solo i bit della porta da PC0 a PC3 sono bit d'ingresso. I bit da PC4 a PC7 sono bit di uscita con questa seconda parola di controllo.

**Passo 5**

Cambiate il byte d'indirizzo LO a LO = 014 in 013 Eseguite il programma con 231 come parola di controllo a LO = 004. Potete concludere che è stato eseguito un latch sull'uscita della porta B?

Sì, dato che qualunque bit a livello logico 1 rimane a tale livello sebbene non vi siano ingressi aggiuntivi all'accumulatore dalla porta C. Viene eseguito un latch solo sulla disposizione iniziale dello switch logico. Tutti i cambiamenti successivi sono ignorati dal programma.

## ESPERIMENTO N. 8

## INTERFACCIAMENTO DI UN CONVERTITORE DIGITALE ANALOGICO

**Scopo**

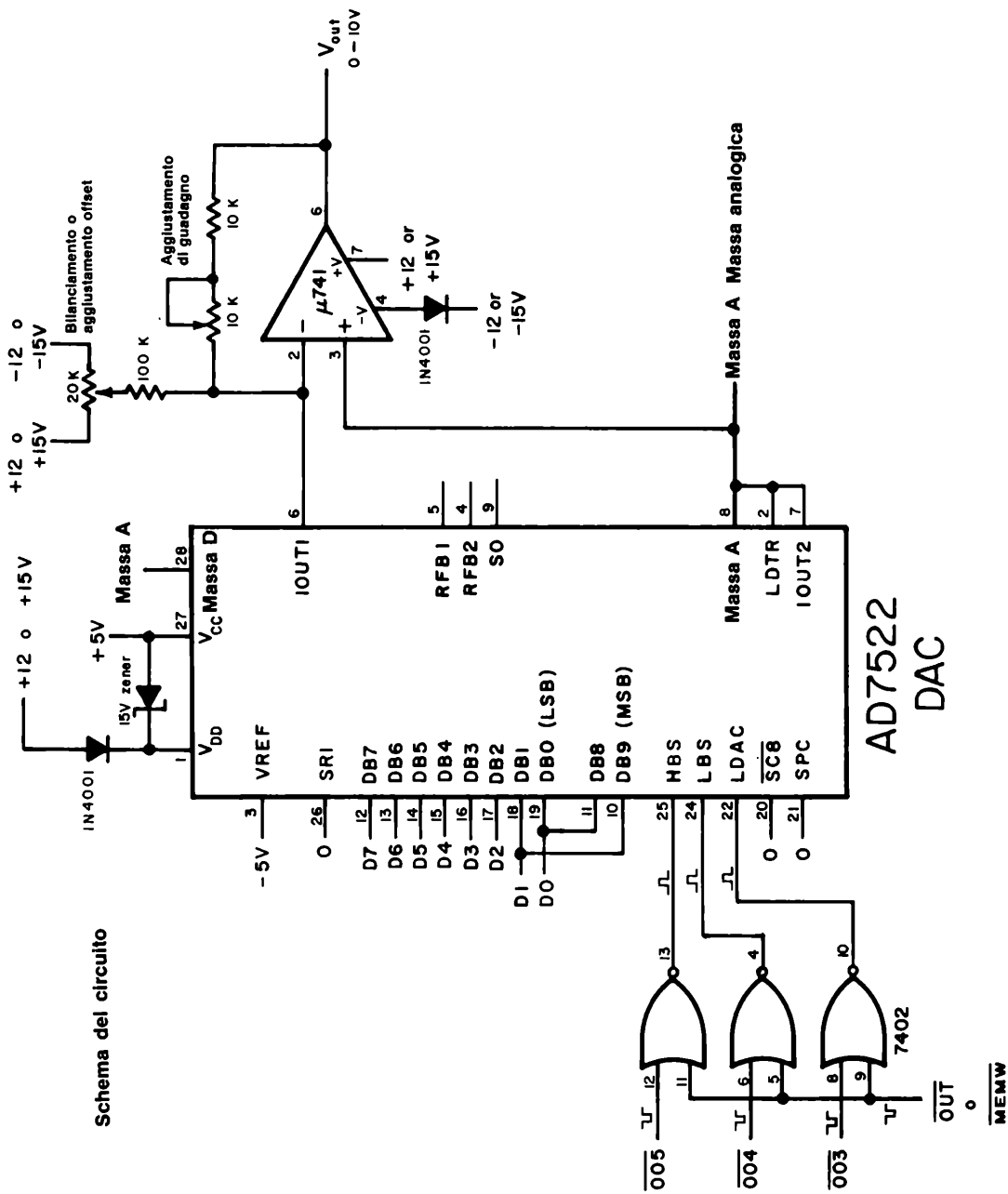
Lo scopo di questo esperimento è provare un semplice programma per un ingresso parallelo dati per il convertitore digitale analogico (D/A) AD7522 con buffer a 10 bit.

**Programma**

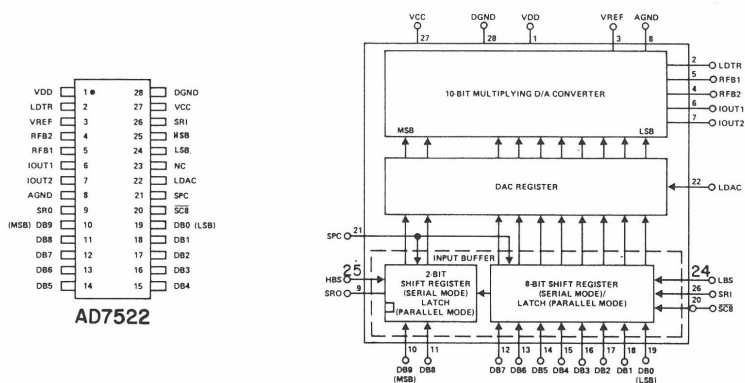
| Indirizzo di memoria LO | Istruzione ottale | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 042 | START, SHLD | Attua lo STROBE di dieci bit di dati digitali verso gli shift register del DAC AD7522 |
| 001 | 004 | 004 | HI = 000 e LO = 004 è il codice dispositivo di I/O memory mapped per l'ingresso LBS del DAC. HI = 000 e LO = 005 è il codice dispositivo di I/O memory mapped per l'ingresso HBS del DAC. (NOTA: possiamo usare questi codici dispositivo dato che il blocco di memoria HI = 000 è EPROM) |
| 002 | 000 | 000 | |
| 003 | 062 | STA | Invia gli impulsi di strobe all'ingresso LDAC del DAC AD7522. Sono forniti segnali di abilitazione a dieci bit di dati digitali all'interno del DAC nel registro del DAC. |
| 004 | 003 | 003 | HI = 000 e LO = 003 è il codice dispositivo di I/O memory mapped per l'ingresso LDAC del DAC |
| 005 | 000 | 000 | |
| 006 | 043 | INX H | Incrementa la coppia di registri H |
| 007 | 315 | CALL | Richiama la routine di ritardo di 10 ms situata nella KEX EPROM (blocco di memoria HI = 000) |
| 010 | 277 | TIMEOUT | Byte d'indirizzo LO di TIMEOUT |
| 011 | 000 | – | Byte d'indirizzo HI di TIMEOUT |
| 012 | 303 | JMP | Salta indietro a START e ripeti l'esecuzione del programma |
| 013 | 000 | START | Byte d'indirizzo LO di START |
| 014 | 003 | – | Byte d'indirizzo HI di START |

**Commento**

Il programma esposto fa sì che il convertitore digitale analogico generi una lenta rampa lineare, che può essere osservata su un voltmetro (VOM) o su un oscilloscopio, come uscita in tensione da DAC. L'uscita della rampa è suddivisa in 1024 piccoli passi, ognuno dei quali è circa 5 - 5,5 mV in d'intensità. Il tempo totale richiesto per cambiare l'uscita da 0 Volt a +5,66 Volt è di 10,24 secondi.

Schema del circuito

La parola del convertitore da digitale ad analogico a 10 bit consiste dei bit di dati da DB0 a DB7, sottoposti a strobe nel registro di shift a 8 bit presente nell'AD7522, con l'aiuto dell'impulso di strobe LBS ed anche dei bit di dati da DB8 a DB9 a cui vengono forniti segnali di abilitazione dall'impulso di strobe HBS. I bit DB8 e DB9 appaiono sul bus di dati bidirezionale del microcomputer come bit D0 e D1.

I particolari degli Analog Devices AD7522 DAC sono descritti nel Bugbook VII - Microcomputer Conversion Devices, che è apparso per la prima volta in USA nell'estate del 1977. Nell'appendice di questo esperimento, forniamo dati aggiuntivi sul DAC AD7522.

**Configurazione dei pin e schema a blocchi del circuito integrato**

**Passo 1**

Studiate lo schema del circuito che cablerete. Quando abbiamo eseguito questo esperimento, $V_{out}$ era collegato ad un piccolo voltmetro (VOM). La massa digitale DGND, dovrebbe essere collegata a quella analogica AGND. Dato che usate le tecniche di I/O memory mapped, $\overline{\text{MEMW}}$ andrebbe usato al posto di $\overline{\text{OUT}}$.

Per facilitare il cablaggio di questo circuito, abbiamo sviluppato un piccolo Outboard, il cui schema è mostrato nella figura seguente. Se avete questo Outboard, cablate il circuito come mostrato. Gli ingressi $\overline{003}$, $\overline{004}$ e $\overline{005}$ sono i canali di uscita decodificati ottenuti da un appropriato circuito di decodifica del byte d'indirizzo LO.

**Passo 2**

Cablate il circuito del DAC e caricate in memoria il programma, mostrato all'inizio di questo esperimento, partendo da HI = 003 e LO = 000. Se state eseguendo questo programma sul microcomputer MMD-1, la routine TIMEOUT di ritardo di 10 ms è già caricata nella KEX EPROM. Se state usando qualche altro microcomputer della famiglia 8080A, nell'appendice dell'esperimento vi forniamo il listing della routine di DELAY.

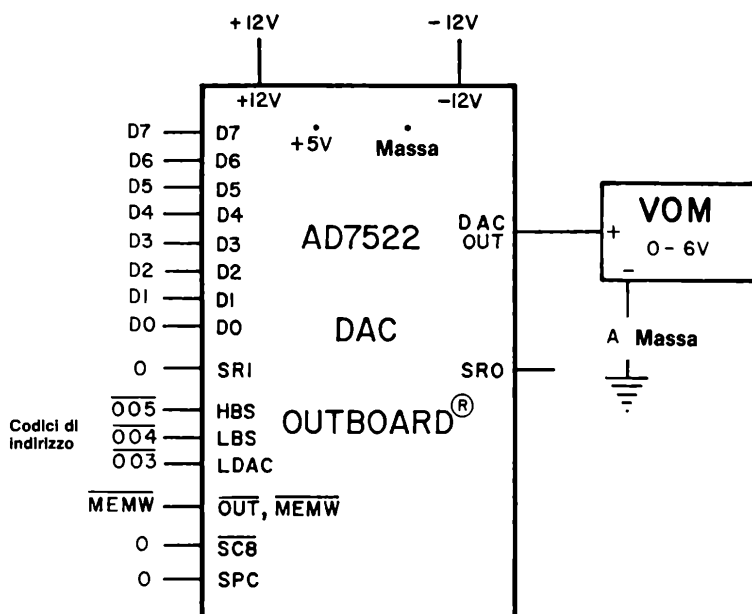



*Figura 22-3.* Schema dell'Outboard DAC AD7522, che contiene tutti i circuiti analogici e digitali necessari per eseguire questo esperimento (vedi schema del circuito).

**Passo 3**

Eseguite il programma con il voltmetro collegato all'uscita del DAC. Che cosa osservate?

Abbiamo osservato un lento ma persistente aumento nella lettura del VOM fino ad arrivare a +5,6 V, momento in cui l'indice è ritornato a 0 V ed ha ripetuto il processo. Il tempo richiesto per tutta la fascia di letture è di circa 10.

**Passo 4**

Cambiate il byte istruzioni a LO = 006 nel seguente:

| | | | |
|---|---|---|---|
| 006 | 053 | DCX H | Decrementa la coppia di registri H |

Eseguite un'altra volta il programma. Che cambiamenti osservate nel comportamento del VOM? Perchè?

Ora il VOM mostra una lenta ma persistente diminuzione da +5,6 a 0 Volt, momento in cui l'indice ritorna a +5,6 Volt e ripete il processo. Decrementiamo il valore invece di incrementarlo come prima.

**Passo 5**

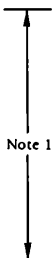
Spostate la subroutine di ritardo, apportando le seguenti modifiche al programma:

| | | | |
|---|---|---|---|
| 007 | 000 | NOP | Nessuna operazione |
| 010 | 000 | NOP | Nessuna operazione |
| 011 | 000 | NOP | Nessuna operazione |

L'istruzione DCX H dovrebbe essere ancora presente. Eseguite questo programma modificato e spiegate che cosa osservate sul VOM.

Abbiamo osservato che l'indice del VOM ha oscillato intorno al valore di +2,75 V. La larghezza delle oscillazioni era di circa +0,02 V. Su di un multimetro digitale, le letture variano da +2,83 a 2,91 V. In altre parole, la rampa lineare piuttosto veloce non potrebbe essere seguita da nessuno dei due strumenti; è stata osservata solo una media.

La rampa lineare era facile da osservare su di un oscilloscopio settato ad un "sweep race" di 10 ms/divisione.

*Conservate il vostro circuito di interfaccia e passate all'esperimento successivo. Informazioni aggiuntive per questo esperimento sono date nell'appendice della pagina seguente.*

**APPENDICE ALL'ESPERIMENTO N. 8**

Segue un listing della routine TIMEUT di ritardo di 10 ms:

| Indirizzo di memoria LO | Istruzione ottale | Codice mnemonico | Descrizione |
|---|---|---|---|
| 277 | 365 | TIMEOUT, PUSH PSW | Inserisci i contenuti dell'accumulatore e dei flag sullo stack |
| 300 | 325 | PUSH D | Inserisci i contenuti della coppia di registri D sullo stack |
| 301 | 021 | LXI D | Carica i due byte seguenti nella coppia di registri D |
| 302 | 046 | 046 | Byte del registro E |
| 303 | 001 | 001 | Byte del registro D |
| 304 | 033 | MORE, DCX D | Decrementa di uno i contenuti della coppia di registri D |
| 305 | 172 | MOV A,D | Sposta i contenuti del registro D nell'accumulatore |
| 306 | 263 | ORA E | OR dei contenuti del registro E con i contenuti dell'accumulatore |
| 307 | 302 | JNZ | Salta a LOOP se il risultato dell'operazione di OR non è 000; altrimenti, salta questa istruzione dopo aver testato il flag di zero |
| 310 | 304 | MORE | Byte d'indirizzo LO di MORE |
| 311 | 000 | – | Byte d'indirizzo HI di MORE |
| 312 | 321 | POP D | Spostamenti dati dallo stack alla coppia di registri D |
| 313 | 341 | POP PSW | Spostamento dati dallo stack all'accumulatore e ai flag |
| 314 | 311 | RET | Rientro dalla subroutine |

Nelle due pagine seguenti forniamo un elenco delle funzioni individuali dei pin sul convertitore digitale analogico AD7522.

# Pin Function Description

| PIN | MNEMONIC | DESCRIPTION |
|---|---|---|
| 1 | VDD | +15V (nominal) Main Supply. |
| 2 | LDTR | R-2R Ladder Termination Resistor. Normally grounded for unipolar operation or terminated at IOUT2 for bipolar operation. |
| 3 | VREF | Reference Voltage Input. Since the AD7522 is a multiplying DAC, VREF may vary over the range of ±10V. |
| 4 | RFB2 | Rfeedback ÷ 2; gives full scale equal to VREF/2. |
| 5 | RFB1 | Rfeedback, used for normal unity gain (at full scale) D/A conversion. |
| 6 | IOUT1 | DAC Current OUT-1 Bus. Normally terminated at virtual ground of output amplifier. |
| 7 | IOUT2 | DAC Current OUT-2 Bus, terminated at ground for unipolar operation, or virtual ground of op amp for bipolar operation. |
| 8 | AGND | Analog Ground. Back gate of DAC N-channel SPDT current steering switches. |
| 9 | SRO | Serial Output. An auxiliary output for recovering data in the input buffer. |
| 10 | DB9 (Note 1) | Data Bit 9. Most significant parallel data input. |
| 11 | DB8 (Note 1) | Data Bit 8. |
| 12 | DB7 (Note 1) | Data Bit 7. |
| 13 | DB6 (Note 1) | Data Bit 6. |
| 14 | DB5 (Note 1) | Data Bit 5. |
| 15 | DB4 (Note 1) | Data Bit 4. |
| 16 | DB3 (Note 1) | Data Bit 3. |
| 17 | DB2 (Note 1) | Data Bit 2. |
| 18 | DB1 (Note 1) | Data Bit 1. |
| 19 | DB0 (Note 1) | Data Bit 0. Least significant parallel data input. |
| 20 | $\overline{SC8}$ | 8-Bit Short Cycle Control. When in serial mode, if $\overline{SC8}$ is held to logic "0," the two least significant input latches in the input buffer are by-passed to provide proper serial loading of 8-bit serial words. If $\overline{SC8}$ is held to logic "1," the AD7522 will accept a 10-bit serial word. Data bits 0(LSB) and DB1 are in a parallel load mode when $\overline{SC8}$ = 0, and should be tied to a logic low state to prevent false data from being loaded. |
| 27 | VCC | Logic Supply. If +5V is applied, all digital inputs/outputs are TTL compatible. If +10V to +15V is applied, digital inputs/outputs are CMOS compatible. |

Note 1: Logic "1" applied to a data bit steers that bit's current to the IOUT1 terminal.

| PIN | MNEMONIC | DESCRIPTION |
|---|---|---|
| 21 | SPC | Serial/Parallel Control. If SPC is a logic "0," the AD7522 will load parallel data appearing on DB0 through DB9 into the input buffer when the appropriate strobe inputs are exercised (see HBS and LBS).<br>If SPC is a logic "1," the AD7522 will load serial data appearing on Pin 26 into the input buffers. Each serial data bit must be "strobed" into the buffer with the HBS and LBS. |
| 22 | LDAC | Load DAC: When LDAC is a logic "0," the AD7522 is in the "hold" mode, and digital activity in the input buffer is locked out. When LDAC is a logic "1," the AD7522 is in the "load" mode, and data in the input buffer loads the DAC register. |
| 23 | NC | No Connection. |
| 25 | HBS | High Byte Strobe. When in "parallel load" mode (SPC = 0), parallel data appearing on the DB9 (MSB) and DB8 data inputs will be "clocked" into the input buffer on the positive going edge of HBS.<br>When in "serial load" mode (SPC = 1), serial data bits appearing at the serial input terminal, Pin 26, will be "clocked" into the input buffer on the positive going edges of HBS and LBS. (HBS and LBS must be clocked simultaneously when in "serial load" mode.) |
| 24 | LBS | Low Byte Strobe. When in "parallel load" mode (SPC = 0), parallel data appearing on the DB0 (LSB) through DB7 inputs will be "clocked" into the input buffer on the positive going edge of the LBS.<br>When in "serial load" mode (SPC = 1), serial data bits appearing at the serial input terminal, Pin 26, will be "clocked" into the input buffer on the positive going edge of HBS and LBS. (HBS and LBS must be clocked simultaneously when in "serial load" mode.) |
| 26 | SRI | Serial Input. |
| 28 | DGND | Digital Ground. |

## ESPERIMENTO N. 9

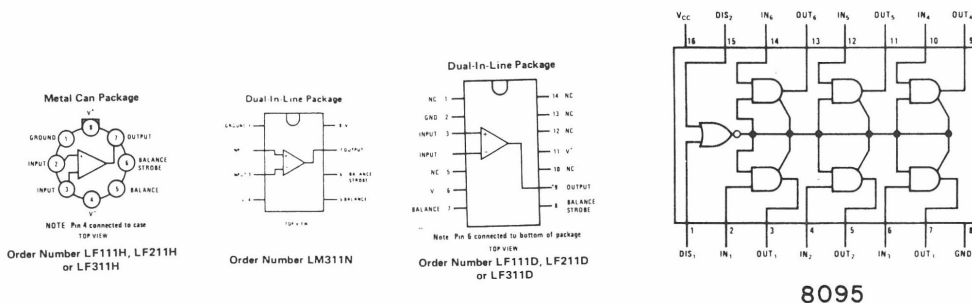
## TECNICA DI COMPARAZIONE "STAIRCASE-RAMP" PER CONVERTIRE UN CONVERTITORE ANALOGICO-DIGITALE

### Scopo

Lo scopo di questo esperimento è usare la tecnica di comparazione *"staircase ramp"* per convertire un convertitore da digitale in analogico AD7522, in un convertitore da analogico in digitale (ADC), con l'aiuto di un comparatore LM311.

### Configurazioni dei pin dei circuiti integrati

### Schema del circuito

Il circuito del DAC è stato mostrato nell'esperimento precedente. L'uscita del DAC, $V_{out}$ dovrebbe ora essere collegata al pin d'ingresso 2 del comparatore LM311, come vedete di seguito.

**Programma**

Questo programma è scritto per essere memorizzato in EPROM, iniziando da HI = 001 e LO = 107 e finendo con HI = 001 e LO = 210. Un asterisco sta ad indicare quelle locazioni di memoria assolute che devono essere cambiate per riposizionare il programma in qualunque punto della memoria. Dato che la maggior parte dei programmatori di EPROM operano in codice esadecimale, anche il programma è scritto così.

| Indirizzo di memoria LO | Istruzione Ott. | Esad. | Codice mnemonico | Descrizione |
|---|---|---|---|---|
| . | | | | |
| . | | | | |
| 107 | 305 | C5 | CONVRT, PUSH B | Salva i contenuti della copp'a di registri B nello stack |
| 110 | 325 | D5 | PUSH D | Salva la coppia di registri D nello stack |
| 111 | 345 | E5 | PUSH H | Salva la coppia di registri H nello stack |
| 112 | 365 | F5 | PUSH PSW | Salva l'accumulatore e i flag nello stack |
| 113 | 323 | D3 | OUT | Genera l'impulso di sincronizzazione |
| 114 | 007 | 07 | 007 | Codice dispositivo dell'impulso di sincronizzazione |
| 115 | 041 | 21 | LXI H | Inizializza la coppia di registri H |
| 116 | 000 | 00 | 000 | Byte del registro L |
| 117 | 000 | 00 | 000 | Byte del registro H |
| 120 | 042 | 22 | AGAIN,SHLD | Attua lo strobe di dieci bit di dati digitali negli shift register AD7522 |
| 121 | 004 | 04 | 004 | HI = 000 e LO = 004 è il codice dispositivo di I/O memory mapped per l'ingresso LBS; HI = 000 e LO = 005 è il codice dispositivo di I/O memory mapped per l'ingresso HBS. |
| 122 | 000 | 00 | 000 | |
| 123 | 062 | 32 | STA | Invia gli impulsi di strobe all'ingresso LDAC del DAC AD7522; carica i dati nel registro DAC |
| 124 | 003 | 03 | 003 | HI = 000 e LO = 003 è il codice dispositivo di I/O memory mapped per l'ingresso LDAC |
| 125 | 000 | 00 | 000 | |
| 126 | 043 | 23 | INX H | Incrementa la coppia di registri H |
| 127 | 333 | DB | IN | Ingresso nel bit D7 dell'uscita del comparatore AD311 |
| 130 | 006 | 06 | 006 | Codice dispositivo per il bit del comparatore |
| 131 | 346 | E6 | ANI | Maschera tutti i bit nell'accumulatore, eccetto il bit D7 |

| | | | | |
|---|---|---|---|---|
| 132 | 200 | 80 | 200 | Byte di maschera |
| 133 | 312 | CA | JZ | Se il bit D7 del comparatore è a livello logico 1, prosegui per i passi successivi; altrimenti, salta ad AGAIN e continua ad incrementare l'uscita del DAC |
| 134 | 120* | 50* | AGAIN | Byte d'indirizzo LO di AGAIN |
| 135 | 001* | 01* | – | Byte d'indirizzo HI di AGAIN |
| 136 | 175 | 7D | MOV A,L | Sposta gli otto bit inferiori della parola del DAC nell'accumulatore |
| 137 | 323 | D3 | OUT | Poni in uscita gli otto bit inferiori della parola del DAC sulla porta di uscita seguente |
| 140 | 002 | 02 | 002 | Codice dispositivo per la porta di uscita 002 |
| 141 | 174 | 7C | MOV A,H | Sposta i due bit più alti della parola del DAC sulla porta di uscita seguente |
| 142 | 323 | D3 | OUT | Poni in uscita i due bit più alti della parola del DAC sulla porta di uscita seguente |
| 143 | 002 | 02 | 000 | Codice dispositivo per la porta di uscita 002 |
| 144 | 361 | F1 | POP PSW | Ripristino dell'accumulatore e dei flag dallo stack |
| 145 | 341 | E1 | POP H | Ripristino della coppia di registri H dallo stack |
| 146 | 321 | D1 | POP D | Ripristino della coppia di registri D dallo stack |
| 147 | 301 | C1 | POP B | Ripristino della coppia di registri B dallo stack |
| 150 | 311 | C9 | RET | Rientro dalla subroutine |

**Commenti**

Osservate che non è difficile riposizionare il programma suddetto. Bisogna cambiare solo due byte d'indirizzo. Il programma genera un'uscita a rampa lineare dal DAC. Questa tensione di uscita $V_{out}$, è continuamente verificata dal comparatore LM311 in opposizione ad una tensione sconosciuta, $E_x$, in questo caso, fornita da un potenziometro da 1 kΩ. Se $V_{out}$ è minore di $E_x$, l'uscita del comparatore è a livello logico 0. Questo unico bit è inserito nella posizione D7 dell'accumulatore, dove viene mascherato ed esaminato dall'istruzione JZ. La rampa lineare continua ad essere generata finchè $V_{out} \geq E_x$, momento in cui l'uscita del comparatore si pone a livello logico 1. L'istruzione JZ viene poi saltata ed avviene un rientro dalla subroutine dopo che la parola DAC a 10 bit è posta in uscita su una coppia di porte di uscita.

**Passo 1**

Supponiamo che il programma suddetto esista già in EPROM. Se non è così, caricatelo in una locazione appropriata nella vostra memoria di lettura/scrittura. I due soli byte d'indirizzo assoluti presenti nel programma sono a LO = 134 e LO = 135.

**Passo 2**

Se il programma è in EPROM, può darsi che dobbiate scrivere un breve programma per richiamare la subroutine. A HI = 003 e LO = 000, caricate il programma seguente nella memoria di lettura/scrittura:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 315 | CALL | Richiama la subroutine CONVRT di comparazione staircase ramp |
| 001 | 107 | CONVRT | Byte d'indirizzo LO di CONVRT |
| 002 | 001 | – | Byte d'indirizzo HI di CONVRT |
| 003 | 166 | HLT | Alt |

**Passo 3**

Osserverete i risultati dell'esecuzione del programma con l'aiuto di un VOM o voltmetro digitale. Il programma nella memoria di lettura/scrittura richiama il programma una volta, permette che venga fatta una sola conversione, e poi si arresta. La tensione analogica misurata appare sul VOM. Nello stesso tempo, il numero del DAC a 10 bit appare sulle due porte di uscita, 000 e 002.

Con il potenziometro da 1 kΩ posizionato su 0 ohm, eseguite una volta il programma. Che cosa osservate?

Abbiamo osservato una lettura di 0,0 V sul nostro voltmetro, una lettura di +0,27 V sul nostro voltmetro digitale (che indica che avevamo degli errori di offset nel nostro circuito analogico), e una lettura di HI = 000 e LO = 000 sulle porte di uscita 000 e 002, rispettivamente.

**Passo 4**

Ora posizionate il potenziometro da 1 K sul suo massimo valore, circa 1 kΩ. Eseguite una volta la routine di conversione ADC, ed osservate sia la tensione di uscita sul VOM e la parola del DAC sulle porte di uscita. Che letture osservate? È quello che vi aspettavate?

Abbiamo osservato una lettura sul VOM di +2,30 V ed una lettura sul nostro multimetro digitale di 2,529 V. Il circuito a divisione di resistenza dovrebbe fornite una lettura di circa +2,8 V, basata sui risultati del nostro precedente esperimento. Le tolleranze del resistore e del potenziometro sono tali che i nostri valori misurati sono ragionevoli. La parola del DAC a 10 bit che appariva sulle porte di uscita 000 e 002 era HI = 001 e LO = 371.

**Passo 5**

Variate la disposizione del potenziometro e provate il programma per la misura delle tensioni fra 0,0 V e +2,5 V. Incontrate delle difficoltà?

No. Le nostre misure sono come ci aspettavamo.

## DOMANDE RIEPILOGATIVE

Le domande seguenti vi aiuteranno a rivedere l'acquisizione dati;

1. Che cos'è un sistema di acquisizione dati?

2. Quali sono le considerazioni importanti nella progettazione di un sistema di acquisizione dati?

3. Come rivelereste un carattere ASCII specifico che viene inserito nel microcomputer?

4. Quali sono i metodi disponibili per generare ritardi di tempo (nuclei di attesa)?

**RISPOSTE**

1. Un sistema di acquisizione dati è uno strumento che automaticamente scandisce i dati prodotti da un altro strumento o elabora e registra letture di dati da usare in seguito.

2. Innanzitutto dovete determinare quanti bit di memoria sono necessari per memorizzare i dati, se tale memorizzare è a breve o a lungo termine, e quanti canali dati al secondo devono essere acquisiti. Non è difficile superare le capacità di un microcomputer 8080A (senza DMA, o accesso diretto in memoria) nelle applicazioni di acquisizione dati a forte velocità.

3. Inserireste il carattere ASCII nell'accumulatore e poi fareste il confronto (CMP) fra i contenuti dell'accumulatore e il codice ASCII specifico del carattere desiderato. Il codice ASCII dovrebbe essere memorizzato nei registri B, C, D, E, H, L o in una locazione di memoria. Quando il carattere desiderato è rilevato, il flag va a livello logico 1.

4. I ritardi di tempo possono essere generati dal software o dall'hardware. Un loop a ritardo di tempo (software) genera ritardi che vanno da venti-trenta microsecondi fino ad ore e perfino a giorni. I metodi hardware per generare ritardi di tempo comprendono l'uso di un clock in tempo reale, operante a 60 Hz, che interrompe l'esecuzione del programma, un clock in tempo reale al quarzo o un interval timer programmabile.

# CAPITOLO 23

# FLAG E INTERRUZIONI

## INTRODUZIONE

L'utilizzo di flag e interruzioni sono tecniche di interfacciamento largamente usate in tutti i tipi di interfaccia per sistemi di elaborazione dati. Questo Capitolo tratta del loro uso con i microcomputer basati sull'8080A e fornisce alcuni esempi tipici hardware e software. Sono discussi anche i problemi inerenti al calcolo dei tempi di interruzione.

## OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Dare la definizione di flag e fare esempi tipici del loro uso.
- Progettare un semplice flag e spiegarne le operazioni.
- Scrivere il software necessario ad uno o più flag e spiegare come si usa tale software.
- Descrivere tre tipi di interruzioni.
- Spiegare l'uso delle istruzioni di ripristino dell'8080A, comprese le operazioni relative allo stack.
- Descrivere l'operatività del sistema di interruzione del microprocessor 8080A e tutti i segnali coinvolti.
- Progettare un sistema d'interruzione e descriverne l'uso.
- Descrivere il software usato in una tipica routine di servizio di una interruzione.
- Spiegare alcuni dei problemi inerenti la tempistica dei flag e delle interruzioni.

## CHE COSA È UN FLAG?

Nei capitoli precedenti abbiamo visto come sia semplice trasferire i dati da e verso un microcomputer usando le istruzioni IN e OUT e un po' di hardware. In molti casi il computer sarà pronto per ricevere i dati in un tempo più breve di quello che occorre alla sorgente di dati per generarli. Può esserci anche il caso di un dispositivo di uscita che può essere molto più lento del computer. Per esempio, una telescrivente può stampare solo da 10 a 30 caratteri al secondo, mentre un tipico sistema 8080A può mettere in uscita un carattere nuovo in 5-10 microsecondi. Chiaramente, occorrono dei metodi di sincronizzazione, in modo che il computer risponda solo quando un dispositivo di ingresso ha effettivamente i dati pronti o quando un dispositivo di uscita ha bisogno di più dati. Ci occorre qualcosa per indicare lo stato del nostro dispositivo: i flag:

*Flag* Un tipo di registro digitale o un dispositivo usato per indicare lo stato di un altro dispositivo. Esso può essere azzerato o settato, in risposta ad un'operazione.

Avete già usato i flag che sono interni al microprocessore 8080A. Essi sono i flag di zero e il flag di riporto che, assieme ai flag di segno, di parità e di riporto ausiliario costituiscono i cinque flag interni dell'8080A, utilizzati nel fare i programmi. Questi flag, escluso il flag di riporto ausiliario, sono la base per il salto o il trasferimento di controllo condizionato. Notate che i flag sono azzerati (livello logico 0) o settati (livello logico 1) in risposta alle vari istruzioni. Questo è conforme alla nostra definizione, dato che il software esegue un'*operazione*, cioè ADD, ROTATE, OR, ecc. È importante notare che i flag si usano per rivelare le varie condizioni e per ricordare poi che condizione si è verificata.

I flag esterni si usano per indicare le condizioni di dispositivi di ingresso/uscita e di altri sistemi digitali o dispositivi che il microcomputer deve controllare. Segue un elenco di alcuni dei tipi di condizioni che vengono indicate tramite l'uso dei flag:

- I dati sono disponibili e letti per essere inseriti nel microcomputer.
- Un dispositivo è pronto perchè successivi otto bit siano posti in uscita verso di esso.
- Un dispositivo esterno è impegnato, o sta ancora eseguendo un'operazione.
- Un dispositivo esterno è pronto per la prossima operazione.
- È stato superato un limite.
- Un valore è troppo basso.

## PRIMO ESEMPIO: COME SI INTERFACCIA UNA TASTIERA

Prendiamo in considerazione un esempio tipico di interfaccia e vediamo come può essere usato un flag. Interfaccieremo una tastiera ASCII a 8 bit al nostro microcomputer progettando una porta di ingresso a 8 bit. Dovreste essere in grado di farlo basandovi sulla vostra esperienza relativa alla decodifica di dispositivi e sulle conoscenze relative alle porte di ingresso three-state. Per ulteriori dettagli, vedere i Capitoli 17 e 20. Il nostro circuito di interfaccia è mostrato nella Figura 23-1.

Nella Figura 23-2 vedete un tipico diagramma di flusso che illustra come inserire i caratteri e confrontarli con la lettera "E".

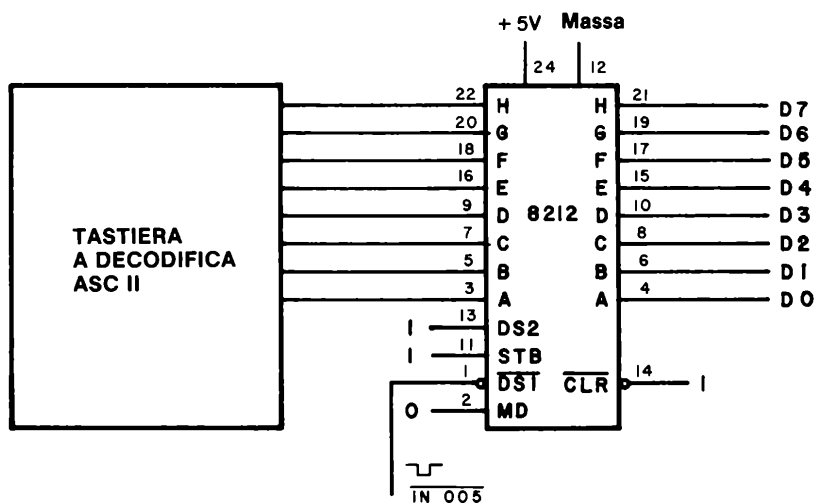



*Figura 23-1.* Tipico circuito d'ingresso ad una tastiera, basato sull'uso di un chip 8212 come porta d'ingresso three-state.

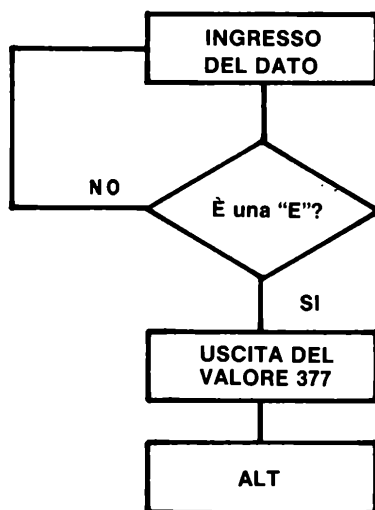



*Figura 23-2.* Diagramma di flusso del sofware necessario alla rivelazione della lettera ASCII, "E".

Quando il codice ASCII per la lettera "E", $305_8$, è finalmente inserito e rivelato dal microcomputer, il programma metterà in uscita tutti 1 logici, o $377_8$, su una porta di uscità, in questo caso il dispositivo 001. Se vi sono dei LED in corrispondenza di questa porta di uscita, essi si accenderanno tutti quando viene rivelata una "E". Il software necessario è il seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | DETECT: IN | Inserisci i dati provenienti dalla tastiera tramite la porta d'ingresso 005. |
| 001 | 005 | 005 | Codice dispositivo 005 |
| 002 | 376 | CPI | Confronta i contenuti dell'accumulatore con il byte ASCII per la lettera "E" |
| 003 | 305 | 305 | Codice ASCII per la lettera "E" |
| 004 | 302 | JNZ | Se il byte d'ingresso della tastiera non è uguale al codice ASCII per la lettera "E", salta alla locazione di memoria DETECT; altrimenti, passa all'istruzione successiva |
| 005 | 000 | 000 | Byte d'indirizzo LO di DETECT |
| 006 | 003 | 003 | Byte d'indirizzo HI di DETECT |
| 007 | 076 | MVI A | I! byte d'ingresso della tastiera è la lettera ASCII "E". Inserisci il byte seguente nell'accumulatore. |
| 010 | 377 | 377 | Byte per l'accumulatore |
| 011 | 323 | OUT | Poni in uscita i contenuti dell'accumulatore, sulla porta di uscita 001. |
| 012 | 001 | 001 | Codice dispositivo 001 |
| 013 | 166 | HLT | Alt |

Questo programma inserirà continuamente i dati dalla tastiera anche quando un tasto non viene premuto (e perciò non vi è nessun codice reale). Noi preferiremmo percepire la pressione dei tasti ed avere l'informazione inserita nel computer quando il codice è valido. La maggior parte delle tastiere forniscono un impulso o un livello che indica quando i dati sono pronti o validi. Questo segnale di stato è un flag. Per esempio, nel caso della nostra tastiera, è un impulso di un microsecondo, chiamato VALID, che indica che è presente un codice valido. Potremmo inserire questo impulso direttamente nel microcomputer e fare una prova per vedere se è presente, ma con tutta probabilità il microcomputer lo perderebbe dato che l'impulso è breve. Abbiamo bisogno di mezzi che prolunghino o trattengano l'impulso fino a che il microcomputer non lo possa utilizzare. La soluzione è un flip-flop che fornisce la possibilità di trattenere l'informazione del flag. Un tipico flag di flip-flop è mostrato nella Fig. 23-3 per entrambi i tipi di flip-flop 7474 e 7476.

L'impulso VALID della tastiera è usato per settare il flag, che può poi essere percepito dal microcomputer sotto il controllo del software.

IN 005

TASTIERA

VALID

Porta di ingresso 8212

Al bus dati

Impulsi di 1 μs

IN 017

D Q

7474

Clock

CLR

D3

Registro degli stati 8212

Al bus dati

OUT 065

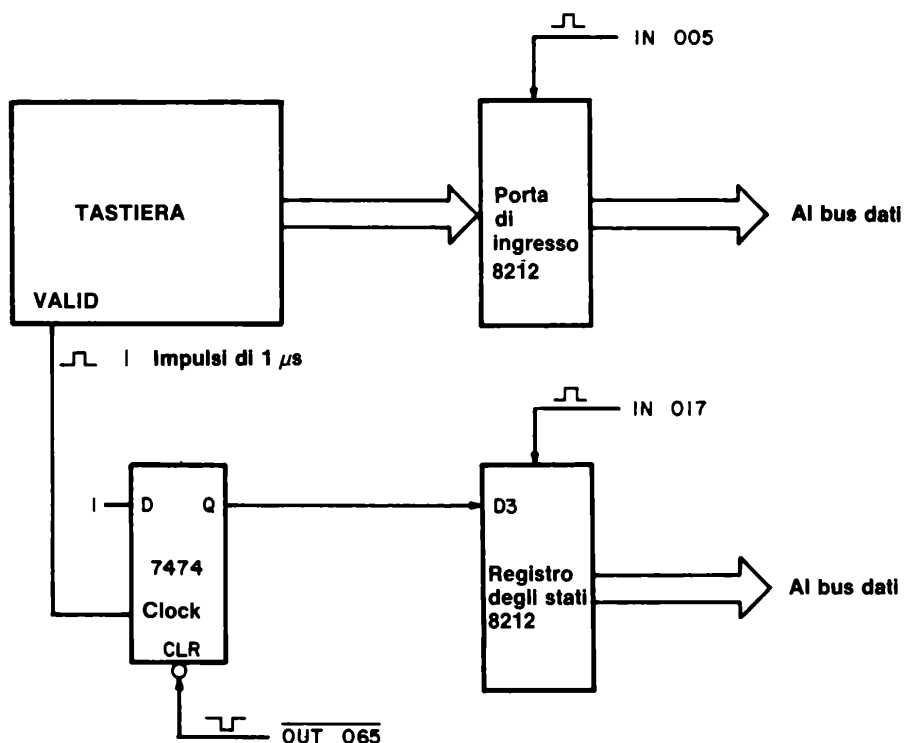



*Figura 23-4.* Circuito semplificato che dimostra come il flag VALID dalla tastiera venga testato dal microcomputer 8080A

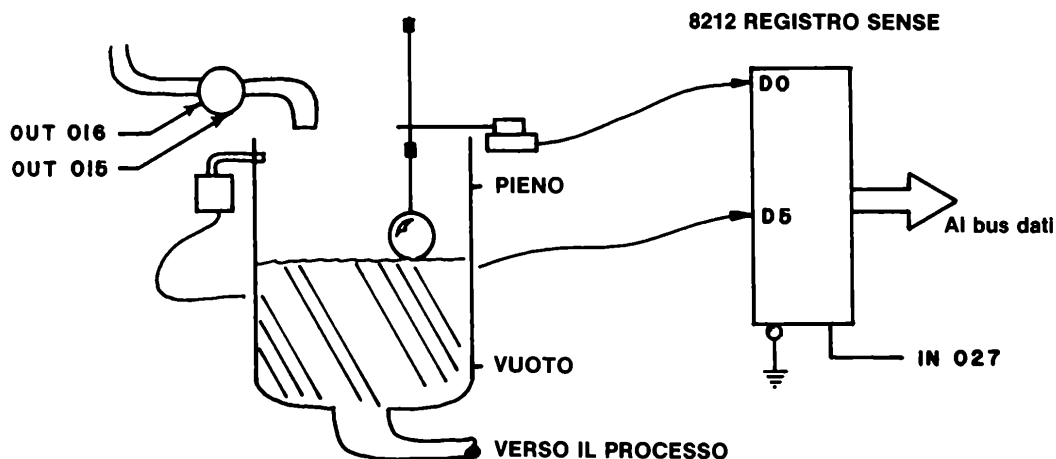



*Figura 23-5.* Circuito che rivela il livello di un liquido con un indicatore di overflow.

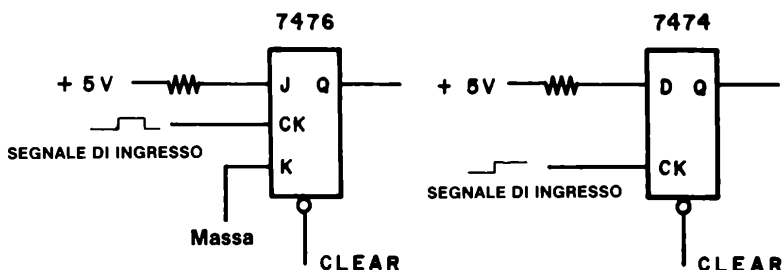



*Figura 23-3.* Tipici flip-flop usati come flag. Il flip-flop 7476 è un dispositivo pulse triggered mentre il flip-flop 7474 è un dispositivo positive edge triggered.

Per inserire l'informazione del flag nell'8080A si usa una porta di ingresso three-state. È esattamente lo stesso tipo di ingresso usato per i dati in ingresso, tranne che ora viene chiamato *registro di lettura,* dato che viene usato con flag individuali. Una volta che è stato letto un flag, esso deve essere azzerato, in modo che la successiva chiusura del tasto possa settare nuovamente il flag. L'hardware è mostrato nella Figura 23-4, nella quale le due porte d'ingresso 8212 sono state semplificate per chiarezza. I decodificatori non si vedono (vedere Capitolo 17 per informazioni sulla generazione di impulsi di selezione dispositivo).

## SECONDO ESEMPIO: CONTROLLO DEL LIVELLO DI UN SOLVENTE

Come secondo esempio dell'uso dei flag, vogliamo controllare il livello del solvente di un serbatoio. Il sistema è rappresentato schematicamente in Fig. 23-5. Si usano due impulsi di selezione dispositivo OUT per controllare la valvola attraverso l'uso di un flip-flop, un buffer, un relè a stato solido, una tecnica già mostrata nella Fig. 17-8 del Capitolo 17. Come precauzione, è stato aggiunto un indicatore di superamento livello di guardia (overflow) che mette in uscita un livello logico 1 quando il solvente sta per traboccare, ed un livello logico 0 quando non vi è pericolo di trabocco. Il commutatore di livello è all'1 logico quando il solvente raggiunge il punto di massimo (pieno) e a stato logico 0 quando raggiunge il punto di minimo (vuoto). Dato che gli indicatori di trabocco e di livello non cambiano rapidamente, possono essere usati direttamente come flag senza flip-flop. Gli indicatori a flip-flop possono essere ancora usati in un ambiente reale; dovreste essere in grado di far vedere come possono essere aggiunti a questo sistema. Un *registro di lettura* three-state viene usato per inserire questi segnali nel microcomputer.

Il vostro scopo è quello di scrivere un programma per mantenere il liquido fra i limiti FULL ef EMPTY e per leggere una condizione di trabocco, che può essere data da un semplice interruttore o da una valvola.
Consideriamo il diagramma di flusso della Fig. 23-6. I flag sono percepiti dal software tramite due istruzioni di salto condizionato. Notate che in questo esempio di diagramma di flusso sono stati usati indirizzi simbolici. Questi simboli o nomi di indirizzi sono usati per semplificare la programmazione dato che gli attuali valori di indirizzo non devono essere assegnati fino a che *il programma non è finito o assemblato.*

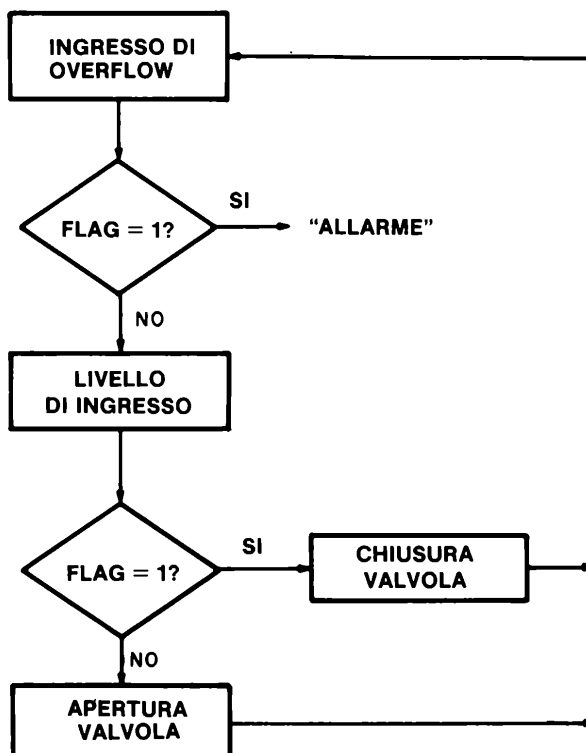



*Figura 23-6.* Diagramma di flusso software per controllare il livello del solvente nel serbatoio.

Il software per il controllo del livello del solvente può essere scritto nel modo seguente:

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 333 | START: IN | Inserisci il valore del flag dati dal dispositivo 027 |
| 001 | 027 | 027 | Codice dispositivo 027 |
| 002 | 346 | ANI | Maschera del bit D5, cioè AND dei contenuti dell'accumulatore con il byte di maschera seguente. |
| 003 | 040 | 040 | Byte di maschera |
| 004 | 304 | CNZ | Se il bit D5 è a stato logico 1, richiama la subroutine ALARM; Altrimenti, passa all'istruzione successiva. |
| 005 | 100 | ALARM | Byte d'indirizzo LO della subroutine ALARM |

| | | | |
|---|---|---|---|
| 006 | 003 | – | Byte d'indirizzo HI della subroutine ALARM |
| 007 | 333 | IN | Inserisci un'altra volta il valore del flag dati dal dispositivo 027 |
| 010 | 027 | 027 | Codice dispositivo 027 |
| 011 | 017 | RRC | Fai ruotare il bit D0 nel flag di carry |
| 012 | 332 | JC | Se il bit di carry è a stato logico 1, il serbatoio è pieno; salta alla locazione di memoria della routine FULL. Altrimenti, passa all'istruzione successiva. |
| 013 | 022 | FULL | Byte d'indirizzo LO della routine FULL |
| 014 | 003 | – | Byte d'indirizzo HI della routine FULL |
| 015 | 323 | OUT | Il serbatoio non è pieno. Apri la valvola e lascia entrare più solvente. |
| 016 | 016 | 016 | Codice dispositivo 016 che genera un impulso di selezione dispositivo che apre la valvola. |
| 017 | 303 | JMP | Ripetere, cioè salta indietro alla locazione di memoria START ed esegui il programma ancora una volta. |
| 020 | 000 | START | Byte d'indirizzo LO di START |
| 021 | 003 | – | Byte d'indirizzo HI di START |
| 022 | 323 | FULL: OUT | Il serbatoio è pieno. Chiudi la valvola che fa entrare il solvente. |
| 023 | 015 | 015 | Codice dispositivo 015 che genera un impulso di selezione dispositivo che chiude la valvola |
| 024 | 303 | JMP | Ripetere, cioè salta indietro alla locazione di memoria START ed esegui il programma ancora una volta. |
| 025 | 000 | START | Byte d'indirizzo LO di START |
| 026 | 003 | – | Byte d'indirizzo HI di START |

Abbiamo supposto che l'indirizzo della subroutine ALARM sia HI = 003 e LO = 100. Il Programma ALARM potrebbe contenere prima un comando per chiudere la valvola, OUT 015, e poi una routine per il segnale di allarme.

Il software suddetto è utile per capire altri aspetti della manipolazione dei bit di flag. Tutti i bit, ad eccezione di D5, sono stati mascherati, o azzerati, nella prima operazione AND, in cui è controllato lo stato di overflow.

Questo significa che l'informazione del flag deve essere inserita di nuovo quando si devono verificare i limiti FULL ed EMPTY. Tale passo potrebbe essere eliminato se i dati di stato fossero conservati in una locazione di memoria, in un registro o nello stack. Ciò può essere importante se gli altri sei bit d'ingresso sono collegati ad altri dispositivi e devono anch'essi essere verificati.

In progetti reali come questo, c'è possibilità di errori. Per esempio, potrebbero essere invertiti i fili dell'interruttore FULL/EMPTY, così nella posizione di pieno la valvola apre e in quella di vuoto chiude. Questo potrebbe essere disastroso; quindi il sistema di controllo di livello del fluido deve essere controllato o simulato prima che inizi l'operazione. Lo farete in uno degli esperimenti. Nel programma di esempio il microcomputer è dedicato ad un piccolo loop che controlla continuamente il serbatoio del solvente. In una situazione reale, verrebbero controllati anche altri serbatoi e livelli. Il tempo per riempire il serbatoio è molto lungo confrontato al tempo in cui il microcomputer può controllare cinquanta o più serbatoi. Se comunque il microcomputer sta eseguendo anche altre operazioni, può darsi che sia in una posizione in cui non può controllare più di una volta ogni serbatoio con un intervallo di parecchi secondi. A seconda degli altri compiti affidatigli, il microcomputer può perdere un livello FULL o anche una condizione OVERFLOW, se occorre troppo tempo per eseguire alcune di queste operazioni.

Un ultimo punto da considerare è il tempo richiesto per aprire e chiudere la valvola. A seconda della valvola, questo tempo può variare da un secondo a cinque o dieci secondi. Nel software di gestione dell'operazione, verrà dato tempo sufficiente perchè la valvola sia aperta o chiusa prima che il microcomputer prenda un'altra decisione sul suo stato. Quando il serbatoio opera vicino alla sua posizione di FULL, dovrebbe essere disponibile una routine per evitare che la valvola di apra o si chiuda se non è necessario.

## OPERAZIONE DI INTERROGAZIONE CICLICA

Il tipo di operazione di cui abbiamo discusso prima, che si usa sia nell'hardware che nel software per la tastiera e per il serbatoio di solvente, è chiamata *operazione di interrogazione ciclica. (Polled operation).* La definizione di *"polling"* è la seguente:

| | |
|---|---|
| *Interrogazione ciclica (Polling)* | Un controllo periodico di ingresso/uscita o dispositivi di controllo per determinare la loro condizione o stato, cioè pieno/vuoto, aperto/chiuso, occupato/libero, fatto/non fatto, ecc. |

Quando i dispositivi vengono interrogati ciclicamente, possono aver bisogno di essere serviti oppure no. Nell'operazione di interrogazione ciclica, i dispositivi sono controllati uno dopo l'altro in sequenza. Quando un dispositivo ha bisogno di essere servito per l'ingresso o l'uscita dei dati o per un'applicazione di controllo, si usa un *"comando software"*, cioè una serie di passi in memoria che sono progettati per servire quel particolare dispositivo. Per esempio, il software per l'ingresso da tastiera e il controllo del serbatoio del solvente dovrebbero essere chiamati comandi di software, dato che determinano un'azione per un particolare dispositivo. Ogni dispositivo di ingresso/uscita ha generalmente una routine di comando da software, o forse anche un insieme di comandi da software per vari tipi di operazioni. Le operazioni di interrogazione ciclica sono di solito lente e vengono usate con dispositivi lenti come telescriventi, lettori di nastro, perforatori, ecc. Per tempi di risposta più brevi, come in un problema a più operazioni, occorre un modo più veloce per servire i dispositivi esterni. Di ciò parleremo più avanti.

## CHE COSA È UN'INTERRUZIONE?

Se foste interrotti mentre leggete queste pagine, finireste probabilmente la frase, vi fareste un segno (forse mentalmente), e poi vi preoccupereste dell'interruzione, cioè una telefonata, il pasto, il bambino; ecc. Dopo aver terminato con la vostra interruzione, continuereste a leggere dal punto in cui siete arrivati. *Le interruzioni in un calcolatore funzionano nello stesso modo!* Il termine, "interrupt" si può definire nel modo seguente:

*Interruzione* — In un computer, una sospensione del normale flusso di un sistema o di una routine tale che il flusso può essere ripreso da quel punto in un secondo tempo.

In un computer, l'operazione di interruzione è molto più sofisticata dell'operazione di interrogazione ciclica ed ha sia vantaggi che svantaggi nei confronti di quest'ultima. Per esempio, nell'operazione di interrogazione ciclica:

- Il computer perde tempo controllando tutti i possibili dispositivi di I/O.
- I dispositivi devono aspettare il loro turno. Tutti sono trattati nello stesso modo, sequenzialmente. Ciò stabilisce una priorità sequenziale, ma ogni dispositivo deve ancora aspettare il suo turno prima di essere servito.
- I tempi di una risposta possono essere lunghi.
- Il software e la programmazione sono generalmente semplici.

Nei sistemi di interruzione:

- Può darsi che il computer stia facendo altre cose non relative ai dispositivi di I/O mentre aspetta che essi richiedano di essere serviti.
- Si può stabilire la priorità nell'hardware o nel software in modo che i dispositivi importanti vengono serviti per primi.
- I tempi di risposta possono essere veloci.
- L'hardware e il software possono diventare molto complessi.

## TIPI DI INTERRUZIONE

Vi sono tre tipi fondamentali di interruzione, *su una sola linea, a più livelli, vettorizzata.*

*Su una sola linea* — Un segnale di interrupt che viene inserito nel computer su una sola linea e fa sì che avvenga un'azione ben definita. Più dispositivi devono essere posti in OR su questa linea. Una routine di interrogazione ciclica deve determinare quale dispositivo ha causato l'interruzione. La famiglia di minicomputer PDP-8 usa questo metodo.

*A più livelli* — Vengono fornite parecchie linee di interrupt indipendenti, ognuna delle quali dà luogo ad un'azione specifica. Non è necessaria l'interrogazione ciclica a meno che più dispositivi non siano in OR su una stessa linea. Il microprocessore 6800 della Motorola usa questo sistema con due linee d'ingresso di interrupt.

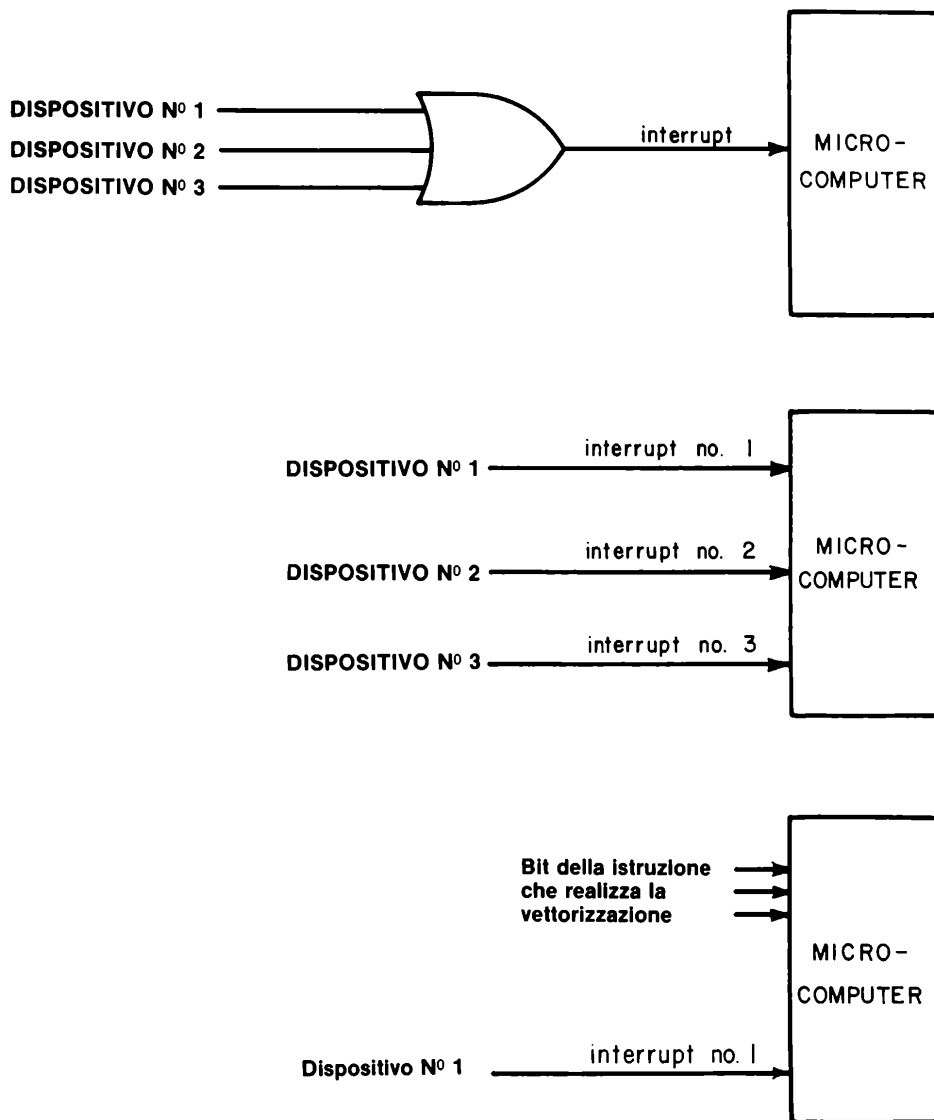



*Figura 23-7.* Schemi che illustrano tre diversi tipi di tecniche di interrupt.

*Vettorizzata* — Ogni dispositivo indica, o segnala con un vettore, lo specifico segnale di controllo per innescare il software specifico per quel dispositivo. La famiglia di minicomputer PDP-11 e l'Intel 8080A usano questa tecnica.

Tutte le tecniche sono mostrate nella Fig. 23-7. Nel sistema su una sola linea, possono essere aggiunti molti dispositivi, ma devono essere tutti interrogati ciclicamente attraverso un registro di lettura usando flip-flop di flag. Il servizio dell'intervento può essere lento, dato che può essere impiegato molto tempo nell'interrogazione ciclica di tutti i dispositivi di un grande sistema. L'interruzione a più livelli è una via di mezzo fra il caso vettorizzato e quello su una sola linea. Ne parleremo dettagliatamente più avanti.

## RIPRISTINO: RST x

In questo Capitolo, ci occuperemo principalmente delle tecniche di interruzione vettorizzata che vengono usate sul microprocessore 8080A. Questo tipo di interruzione ci permette di fornire non solo un impulso di interrupt, ma anche un'istruzione al microprocessore per dirgli che cosa fare. Nei sistemi 8080A semplici, un'istruzione ad un solo byte può essere "forzata" nel computer quando esso viene interrotto. Mentre le istruzioni di rotazione, di incremento ed altre istruzioni ad un solo byte potrebbero essere utili in alcune applicazioni, le istruzioni di *ripristino,* cioè le istruzioni di richiamo di subroutine, ad un solo byte, sono molto più utili e flessibili.

Le istruzioni di richiamo dell'8080A, sia condizionate che incondizionate, specificano ognuna un indirizzo a sedici bit nel secondo (LO) e nel terzo (HI) byte istruzioni. Come può esserci un richiamo di subroutine ad un solo byte? La risposta è che le istruzioni di ripristino richiamano le subroutine a indirizzi predefiniti. Ecco l'elenco di queste istruzioni:

307 RST 0 Richiama la subroutine a HI = $000_8$ e LO = $000_8$
317 RST 1 Richiama la subroutine a HI = $000_8$ e LO = $010_8$
327 RST 2 Richiama la subroutine a HI = $000_8$ e LO = $020_8$
337 RST 3 Richiama la subroutine a HI = $000_8$ e LO = $030_8$
347 RST 4 Richiama la subroutine a HI = $000_8$ e LO = $040_8$
357 RST 5 Richiama la subroutine a HI = $000_8$ e LO = $050_8$
367 RST 6 Richiama la subroutine a HI = $000_8$ e LO = $060_8$
377 RST 7 Richiama la subroutine a HI = $000_8$ e LO = $070_8$

che possono essere riassunte nel modo seguente;
3X7 RST X Richiama la subroutine a HI = $000_8$ e LO = $0X0_8$

Le locazioni della subroutine, HI = $000_8$ e LO = $0X0_8$, sono prestabilite nell'8080A e non possono essere cambiate. Vi sono altri modi di superare questa limitazione se volete usare altre locazioni per il software di gestione dell'interruzione.

L'istruzione di ripristino 3X7 è forzata nell'8080A solo durante un'interruzione. Come con altri ingressi, quali lettura di memoria e ingresso di I/O tramite l'accumulatore, i dati per il byte istruzioni RST X devono esssere sottoposti a gate sul bus dell'8080A, al momento opportuno. Vinene fornito un segnale aggiuntivo, *interrupt acknowledge* (INTA o IACK), per sincronizzare l'ingresso dell'istruzione ad un byte. Tale segnale è usato per fornire segnali di abilitazione al byte istruzione sul bus dati e nel chip 8080A. Il byte istruzione va direttamente al *registro istruzione* e non a qualcuno dei registri universali. Il flusso del segnale di interrupt è mostrato schematicamente in Fig. 23-8.

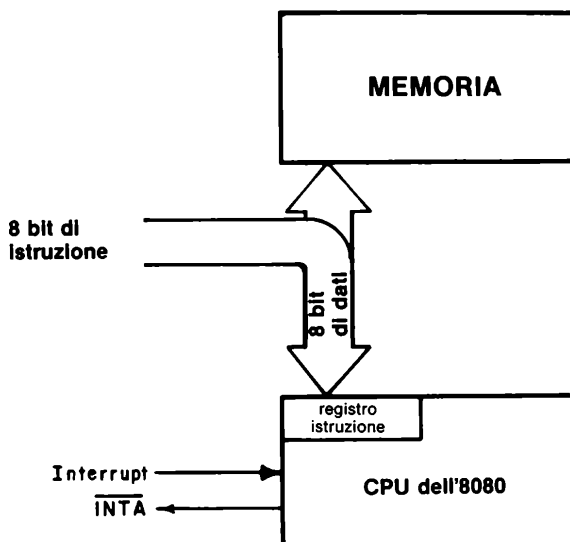



*Figura 23-8.* Flusso del segnale di interrupt per un tipico microcomputer 8080A. L'istruzione a otto bit è posta, tramite gate, sul bus di dati da $\overline{\text{INTA}}$.

Per inserire l'istruzione di interrupt si usa una porta d'ingresso three-state standard ottenuta per chip quali il buffer/latch 8212 o il DM8095 (74365), usando $\overline{\text{INTA}}$ invece di $\overline{\text{IN 006}}$ o alcuni altri impulsi di selezione dispositivo come impulso di abilitazione.

## ABILITAZIONE E DISABILITAZIONE DELL'INTERRUZIONE: EI e DI

L'8080A ed altri microprocessori hanno una caratteristica molto utile: la CPU è in grado di rendersi immune dalle richieste di interruzione esterne. Possiamo attivare le interruzioni, per far sì che vengano accettare, o disattivarle e ignorarle. Vi sono momenti in cui vogliamo che l'interruzione non venga usata affatto. Quando l'8080A inizia ad operare o è resettato, esso disattiva le interruzioni. È responsabilità del programmatore abilitare l'interruzione se il chip 8080A è fatto per accettare e servire le interruzioni. Questo avviene tramite un'istruzione software, *"enable interrupt"* o EI. Possiamo eseguire anche l'operazione complementare, *"disable interrupt"* o DI.

| | | |
|---|---|---|
| 373 | EI | Abilita il sistema d'interruzione ed accetta le interruzioni dopo l'esecuzione dell'istruzione *seguente* |
| 363 | DI | Disabilita il sistema d'interruzione e rifuita altre interrruzioni. Ciò avviene immediatamente. |

La capacità di interruzione può essere abilitata solo sotto il controllo del software.

Possiamo pensare al processo di abilitazione/disabilitazione come ad un processo interno di un flag dell'8080A. Se il flag è abilitato, le interruzioni sono accettate attraverso la sezione di controllo del chip 8080A. Quando il flag è disabilitato, le interruzioni sono bloccate.

L'ingresso di interrupt va ad un altro flag interno dell'8080A, che può ricordare il verificarsi dell'interruzione. Una terza uscita di controllo, *interrupt enable* (INTE), che è il pin 16 sull'8080A, può essere usata per indicare ai dispositivi esterni e alle interfacce che l'interruzione è abilitata (livello logico 1) o disabilitata (livello logico 0). L'interruzione è sempre disabilitata dopo che si accetta un'interruzione da un dispositivo esterno.

## TERZO ESEMPIO: APPLICAZIONE DELL'INTERRUZIONE ALL'INTERFACCIA DI UNA TASTIERA

Diamo un altro sguardo all'interfacciamento della tastiera per vedere come si può usare un'interruzione al posto di un flag. In alcune applicazioni, per esempio, dove la tastiera ed un certo numero di altri dispositivi di I/O sono collegati ad un computer, il computer stesso può impiegare molto tempo a ritornare indietro e ad interrogare ciclicamente la tastiera. I caratteri possono essere persi o ignorati se il software non è scritto attentamente. La soluzione di questo problema è l'interruzione, che fornisce un servizio quasi immediato ai dispositivi esterni. Per usare con successo l'interruzione abbiamo bisogno di collegare l'impulso di uscita VALID della tastiera all'ingresso di interrupt dell'8080A al pin 14. L'uscita VALID è un impulso positivo, come richiesto dal chip 8080A, così non è necessaria nessuna inversione o intervento di un buffer. Dobbiamo anche fornire l'istruzione di ripristino, in questo caso RST 5, che ha un codice istruzione di 357. Questo byte istruzione è mandato direttamente al registro istruzione dell'8080A quando si manifesta l'interruzione. Con l'aiuto di un buffer/latch 8212, possiamo inserire questo byte istruzione nell'hardware ad un *registro istruzione di interrupt o ad una porta istruzione di interruzione,* come si vede in Fig. 23-9.

Rivediamo velocemente l'operazione di un'interruzione. Per prima cosa, il flag di interrupt deve essere abilitato all'interno dell'8080A, usando l'istruzione EI. Poi, un segnale esterno causa un'interruzione del'8080A la riconosce, generando il segnale di riconoscimento d'interruzione, $\overline{\text{INTA}}$, che è usato per inserire con un gate un'istruzione di ripristino ad un byte nel registro istruzione dell'8080A. Usiamo un'istruzione di ripristino, precisamente RST 5, per richiamare la subroutine di servizio a HI = 000 e LO = 050, la locazione di memoria dove inizia il software per l'ingrsso della tastiera.

L'interruzione della tastiera agisce per inserire una routine specifica nel normale flusso del software. Segue un esempio tipico di come questo accade.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 000 | 061 | LXI SP | Carica lo stack pointer con i due byte seguenti |
| 001 | 000 | 000 | Byte LO dello stack pointer |
| 002 | 004 | 004 | Byte HI dello stack pointer |
| 003 | 373 | EI | Abilita l'interruzione |
| 004 | -- | MAIN | Questi passi comprendono il MAIN TASK (compito principale) del programma |
| 005 | -- | TASK | |

| | | | |
|---|---|---|---|
| 050 | 333 | IN | Inserisci i dati provenienti dalla tastiera nella porta d'ingresso 005 |
| 051 | 005 | 005 | Codice dispositivo 005 |
| 052 | –– | ALTRO SOFTWARE DI SERVIZIO INTERRUZIONE | |
| 053 | –– | | |
| . | | | |
| . | | | |
| . | | | |
| –– | 311 | RET | Rientro dalla subroutine |

*Figura 23-9.* Circuito di interruzione vettorizzata semplificato, per una tastiera a caratteri ASCII. Se vi sono altri dispositivi d'interruzione nel sistema, va usato un gate NAND prima dell'ingresso INT sull'8080A.

Perchè sono state incluse le istruzioni LXI SP e RET? Ricordate che le istruzioni di ripristino sono istruzioni di richiamo ad un solo byte che richiamano le subroutine ad indirizzi specifici. Perciò, quando si usano tali istruzioni, gli indirizzi di ritorno o di rientro saranno ancora memorizzati sullo stack.

Le istruzioni di ripristino pur essendo molto utili per le interruzioni, sono pur tuttavia istruzioni dell'8080A valide per un normale uso in programmi. Se volete impiegare una subroutine ad uno degli indirizzi di vettore, HI = 000 e LO = 0X0, usate un'istruzione di ripristino per richiamarla. Il programma suddetto lavorerà se lo mettete in esecuzione, ma risponderà solo alla prima pressione di tasti, o interruzione. Perchè solo una pressione? La ragione è che in qualunque momento venga abilitato il flag di interrupt ed accetti un'interruzione, il flag di interrupt diventa immediatamente disabilitato e non accetta ulteriori interruzioni. Questo impedisce che il software di servizio dell'interruzione venga riinterrotto immediatamente. L'8080A non accetterà ulteriori interruzioni fino a che il flag di interrupt non è riabilitato con un'istruzione di abilitazione all'interruzione, o EI. Nell'esempio della tastiera, non vi è nessuna istruzione di abilitazione all'interruzione nel MIN TASK o nella subroutine della tastiera, così il flag non può essere riabilitato.

Per riabilitare il flag di interrupt, bisognerebbe mettere un'istruzione di abilitazione all'interruzione, immediatamente prima del byte istruzione RET. Così, il controllo di programma può perlomeno ritornare al MAIN TASK prima che l'8080A accetti un'altra interruzione. Perché questo è importante? Se un dispositivo d'interruzione potesse interrompere immediatamente dopo l'istruzione EI, il chip 8080A accetterebbe l'interruzione e l'istruzione di rientro non sarebbe eseguita. Se il chip 8080A permettesse che questo avvenisse molte volte, lo stack potrebbe riempirsi di indirizzi di rientro (dato che essi non verrebbero estratti dallo stack e usati dalle istruzioni di rientro). Ecco perchè è importante che le interruzioni siano accettate solo dopo l'esecuzione dell'istruzione che segue EI. L'esecuzione del rientro ci permette di "ripulire" lo stack dopo che è finita una subroutine di interrupt.

Trattiamo le nostre subroutine "vettorizzate" come se fossero subroutine normali, cioè le istruzioni PUSH e POP possono essere usate per memorizzare e recuperare i dati.

*Figura 23-10.* Una tipica subroutine di servizio dell'interruzione. Le prime istruzioni, le istruzioni PUSH, conservano lo stato del microcomputer. Vicino alla fine della subroutine, lo stato del microcomputer viene ripristinato nei registri interni.

Nella Figura 23-10 vedete come apparirebbe una tipica subroutine di interrupt. Dato che vi sono solo otto posizioni fra l'indirizzo vettorizzato della tastiera, 050, e l'indirizzo vettorizzato seguente, 060, come può esser usato tutto questo software? Se 060 viene usato come indirizzo vettorizzato per un altro dispositivo, abbiamo certamente un problema! Possiamo aggirare il problema mettendo semplicemente un'istruzione JMP, a tre byte, nelle locazioni 050, 051 e 052, che trasferisca il controllo del programma in un'area di memoria dove c'è più posto per il software. Il prezzo che dobbiamo pagare per questo è un ritardo di 10 stati di clock, cioè 5 microsecondi per un microcomputer a 2 MHz e 13,33 microsecondi per un microcomputer a 750 kHz. L'istruzione RET alla fine della routine di servizio, rimanda ancora il programma di controllo al punto in cui il MAIN TASK si era interrotto ed era stata eseguita l'istruzione RST 5, come indicato schematicamente nella Figura 23-11.

Avremmo potuto fare cose notevolmente più complicate inserendo interruzioni differite e interruzioni di priorità, ma questi sono argomenti complessi che vanno oltre lo scopo del nostro semplice esempio riguardante la tastiera.

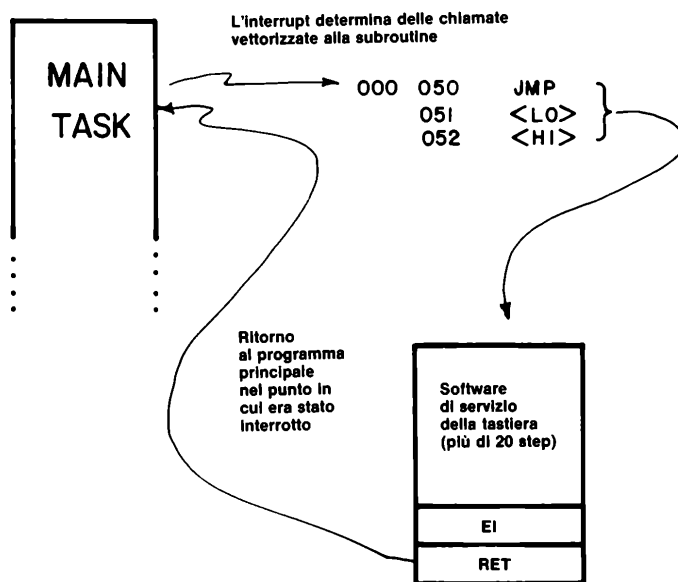



*Figura 23-11.* Relazioni fra il MAIN TASK, il salto alla subroutine vettorizzata, e il software di servizio della tastiera, che è posto in un punto della memoria.

### INTERRUZIONI CON PRIORITA'

Le *interruzioni di priorità* sono interruzioni che vengono messe in ordine di importanza cosicchè alcuni dispositivi d'interruzione hanno la precedenza su altri. Quando avviene un certo numero di interruzioni contemporaneamente, o quando esiste questa possibilità, abbiamo bisogno di metodi per determinare quale dispositivo dovrebbe essere servito e in quale ordine. Nel circuito mostrato nella Fig. 23-12, vengono mostrate per chiarezza tre interruzioni, ma potrebbero esserne aggiunte facilmente altre. Un'interruzione avviene in qualunque momento e uno dei flip-flop di flag viene settato da un impulso applicato al suo ingresso di clock. Quando avviene l'interruzione, l'8080A genera un impulso $\overline{\text{INTA}}$ ed pone in ingresso il codice dell'istruzione di ripristino, 357, che è precedentemente cablato alla porta per l'istruzione di interrupt dell'8212. L'istruzione 357 vi manda direttamente all'indirizzo di memoria HI = 000 e LO = 050, dove inizia il software per l'interrogazione ciclica dei dispositivi d'interruzione. A questo punto dovreste aver ben capito le istruzioni relative alla vettorizzazione ed al ripristino. Parleremo ora della routine di interrogazione ciclica, che è mostrata di seguito.

| Indirizzo di memoria LO | Byte di istruzione | Codice mnemonico | Descrizione |
|---|---|---|---|
| 050 | 333 | POLL: IN | Inserisci il bit di stato |
| 051 | 057 | 057 | Codice dispositivo 057, per il registro di lettura |
| 052 | 057 | CMA | Complementa i bit di stato nell'accumulatore (1→0 e 0→1) |
| 053 | 346 | ANI | Maschera tutti i bit meno i bit D0, D1 e D2 |
| 054 | 007 | 007 | Byte di maschera |
| 055 | 037 | RAR | Rotazione del bit D0 nel flag di Carry |
| 056 | 332 | JC | Se il flag di carry è a stato logico 1, salta alla routine di servizio della cassetta CASSVC |
| 057 | 100 | CASSVC | Byte d'indirizzo LO di CASSVC |
| 060 | 003 | – | Byte d'indirizzo HI di CASSVC |
| 061 | 037 | RAR | Rotazione del bit d'ingresso D1 nel flag di carry |
| 062 | 332 | JC | Se il flag di carry è a stato logico 1, salta alla routine di servizio della tastiera KBRD. |
| 063 | 200 | KBRD | Byte d'indirizzo LO di KBRD |
| 064 | 003 | – | Byte d'indirizzo HI di KBRD |
| 065 | 037 | RAR | Routine del bit d'ingresso D2 nel flag di carry. |
| 066 | 332 | JC | Se il flag di carry è allo stato logico 1, salta alla routine di servizio clock di un'ora, CLOCK. |

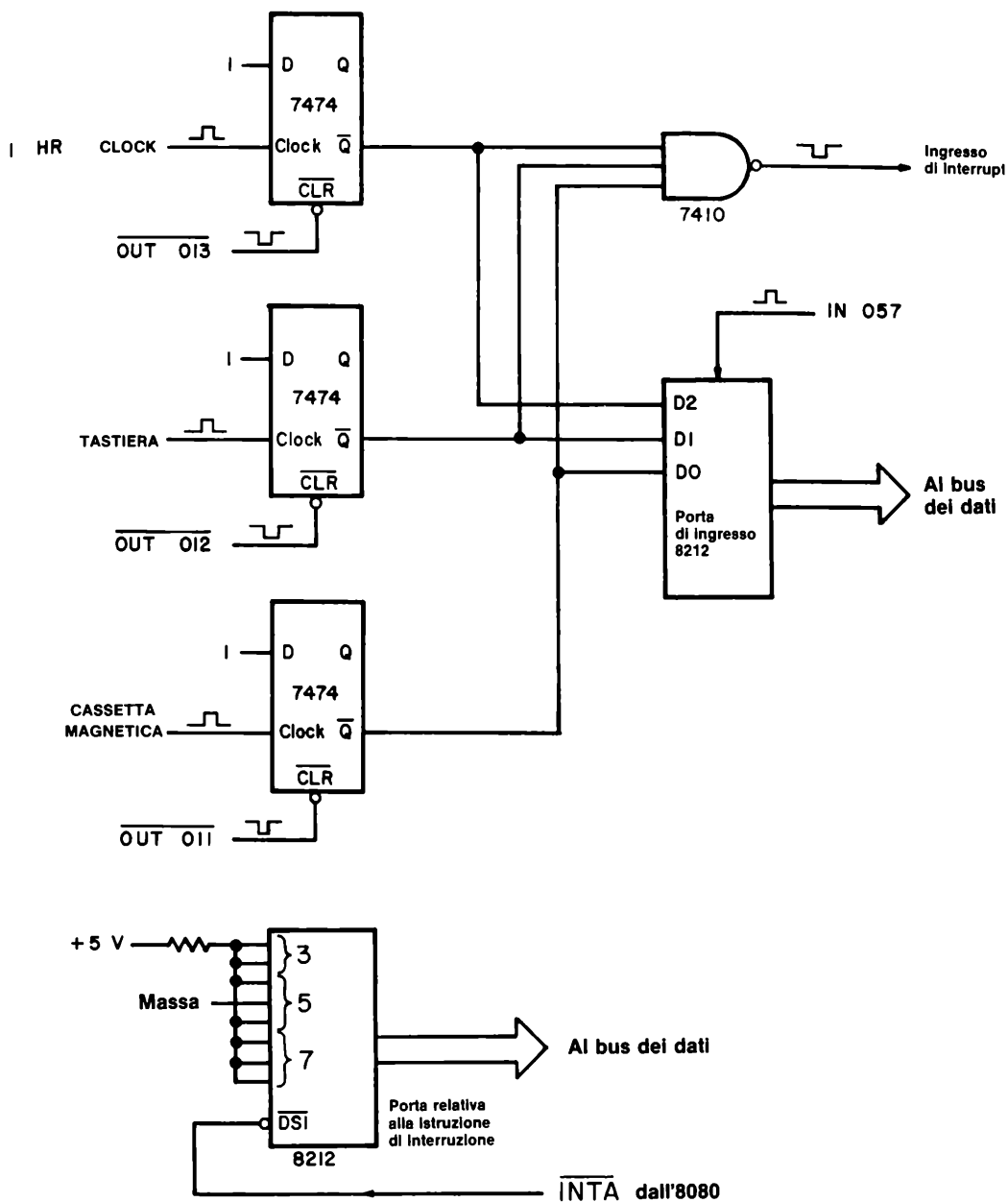



*Figura 23-12.* Circuito di interrupt ad interrogazione ciclica: consiste di tre dispositivi di interrupt e di un ingresso vettorizzato RST.

| | | | |
|---|---|---|---|
| 067 | 300 | CLOCK | Byte d'indirizzo LO di CLOCK |
| 070 | 003 | – | Byte d'indirizzo HI di CLOCK |
| 071 | 166 | HLT | Alt. Se arrivate a questo punto, il programma viene interrutto, ma non da uno dei tre dispositivi suddetti. |

Ogni bit di flag è ad 1 se non è necessario il servizio e a 0 se il servizo è necessario. Il gate NAND a tre ingressi fornisce un livello logico 1 al microcomputer 8080A quando qualunque dispositivo genera un'interruzione; questo stato logico è inserito nel pin 14 dell'8080A. Viene stabilita la priorità in modo da servire prima la cassetta (dispositivo a grande velocità), segue poi la tastiera (dispositivo a velocità ridotta) e per ultimo il clock da un'ora (dispositivo molto lento). Come per i dati d'ingresso dell'accumulatore, preferiamo avere un livello logico 1 se è necessario il servizio, ed un livello logico 0 se il servizio non è necessario. Il nostro primo passo di programma dovrebbe quindi invertire, o complementare, i dati in ingresso della parola dei flag con l'uso di un'istruzione CMA a LO = 052. Questo semplice passo di programma illustra come è facile invertire i dati dell'accumulatore ed eliminare tre invertitori 7404 o ancora eliminare le necessità di modificare l'hardware in modo che l'uscita Q, invece dell'uscita $\overline{Q}$, venga inserita sul gate 7410.
Dopo l'istruzione CMA, mascheriamo tutti gli altri bit dei dispositivi meno i bit D0, D1 e D2. Facciamo poi ruotare questi bit nel carry e testiamo se sono ad uno stato logico 1, che indica che un dispositivo specifico ha generato un'interruzione. Ogni routine di servizio CASSVC, KBRD, CLOCK - è molto simile alla routine di servizio dell'interruzione mostrata in Fig. 23-10, e termina con l'abilitazione all'interruzione, EI e con le istruzioni di rientro, RET.
Vanno date altre spiegazioni sulla routine di interrogazione ciclica. Benchè essa passi attraverso gli indirizzi "di vettore" 060 e 070, in questo caso non è un errore, dato che non abbiamo altre interruzioni che li usano. Probabilmente inizieremmo la nostra routine di interrogazione ciclica con un'istruzione PUSH PSW, *dato che non sappiamo per quale scopo il MAIN TASK ha usato l'accumulatore ed i flag, quando è stato interrotto.*
Se usassimo PUSH PSW, ogni routine di servizio richiederebbe una POP PSW immediatamente prima dell'istruzione di abilitazione all'interruzione. La prima cosa che dovremmo fare in ogni *routine di servizio* è azzerare il flag associato con il dispositivo d'interruzione. L'istruzione OUT va bene per generare impulsi che azzerano i flip-flop, come mostra la Figura 23-12. Perciò, un'istruzione OUT 011 azzera il flag della cassetta, una OUT 012 azzera il flag della tastiera e una OUT 013 azzera il flag del clock da un'ora. Vanno bene anche altri schemi di software a interrogazione ciclica ed altri metodi di test dei bit; quello che vi abbiamo mostrato è semplice e vantaggioso.
Il clock da un'ora solleva un'importante questione: perchè si dovrebbe costruire un clock hardware esterno da un'ora quando il software può farlo sotto i 50 byte? La risposta dipende da come usate il vostro microcomputer. Se il microcomputer può eseguire il loop di software da un'ora o se state usando le interruzioni e potete tollerare errori, usate il software. Se avete bisogno di un tempo esatto e state usando le interruzioni, usate l'hardware. Come arrivare a questa decisione? Quando usate le interruzioni, aggiungete del software nel flusso di programma MAIN TASK. Tutto questo prende tempo, dato che il software deve non solo controllare il dispositivo d'interruzione, ma deve anche servirlo. Se interrompete la routine software da un'ora per cinque volte con una routine di servizio di un dato dispositivo, che impiega due minuti per l'esecuzione, avete realmente impiegato un'ora e dieci minuti per raggiungere il vostro scopo, cioè le operazioni software di un'ora sono sospese quando interrompete ed eseguite un altro compito. Il tempo reale va avanti, mentre il tempo del software è sospeso. Il clock esterno da un'ora potrebbe essere chiamato *clock a tempo reale,* dato che esso impiega il tempo reale, *non* il tempo del software del calcolatore!

## INTERRUZIONI CON PRIORITA' DA HARDWARE

Oltre che con interruzioni a interrogazione ciclica, si può dare priorità alle interruzioni anche usando l'hardware. Questo tipo di interruzione con priorità è importante, ogni volta che un certo numero di dispositivi d'interruzione (che richiedono tutti un servizio veloce) sono collegati ad un microcomputer. Ogni dispositivo genera la sua istruzione di ripristino, RST X, che, una volta accettata, dà luogo ad una vettorizzazione immediata alla locazione di memoria HI = 000 e LO = OXO. La priorità è assegnata tramite l'uso di un codificatore di priorità 74148, che accetta fino ad otto ingressi di flag, ognuno a livello logico 0, se esiste una condizione di interruzione per ogni dispositivo, *e mette in uscita il codice binario a tre bit per l'ingresso con il numero più alto che è a livello logico 0.* La Figura 23-13 mostra una tabella della verità e lo schema del chip. Il 74148 è usato in combinazione con una regolare istruzione d'interruzione nell'ambito dell'hardware relativo alla interruzione con priorità, come mostra la Figura 23-14.

SN54148, SN74148
FUNCTION TABLE

| INPUTS | | | | | | | | | OUTPUTS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EI | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | A2 | A1 | A0 | GS | EO |
| H | X | X | X | X | X | X | X | X | H | H | H | H | H |
| L | H | H | H | H | H | H | H | H | H | H | H | H | L |
| L | X | X | X | X | X | X | X | L | L | L | L | L | H |
| L | X | X | X | X | X | X | L | H | L | L | H | L | H |
| L | X | X | X | X | X | L | H | H | L | H | L | L | H |
| L | X | X | X | X | L | H | H | H | L | H | H | L | H |
| L | X | X | X | L | H | H | H | H | H | L | L | L | H |
| L | X | X | L | H | H | H | H | H | H | L | H | L | H |
| L | X | L | H | H | H | H | H | H | H | H | L | L | H |
| L | L | H | H | H | H | H | H | H | H | H | H | L | H |

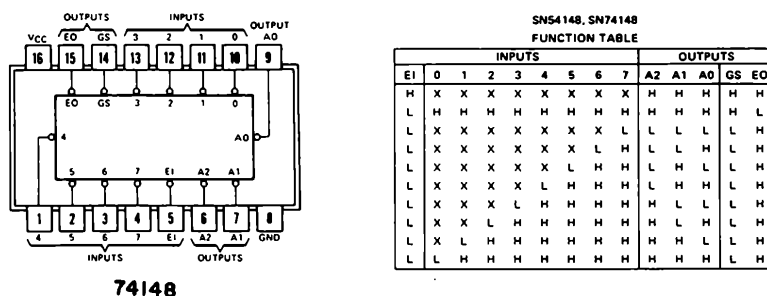



*Figura 23-13.* Configurazione dei pin e tabella della verità del codificatore di priorità 74148 da 8 a 3 linee.

Se le richieste di interruzione simultanea sono generate dai dispositivi 5 e 7, il dispositivo 7 ha la priorità massima e il chip 74148 e gli invertitori della Figura 23-14 forniscono un "7" per l'istruzione 3X7. Questa dà luogo ad un riferimento vettorizzato alla locazione di memoria HI = 000 e LO = 070. Mentre abbiamo a disposizione un vettore RST 0, non lo usiamo spesso dato che il suo unico effetto è quello di resettare il contatore di programma e far incominciare di nuovo il programma MAIN TASK.

Nella Figura 23-14 non vengono mostrati, per chiarezza, i flag necessari e la disposizione dei flag stessi o le linee di azzeramento. Ricordatevi comunque che *andrebbe usato un flag per ogni dispositivo d'interruzione.* Si potrebbero aggiungere altri particolari hardware al circuito per renderlo più efficiente. Fra questi potrebbe esserci un decodificatore addizionale per generare l'impulso di azzeramento del flag senza bisogno di un'istruzione OUT, ed un registro maschera in modo da mascherare parecchi dispositivi nell'hardware esterno. Tali aggiunte sono mostrate in Figura 23-15, in uno schema di interruzione a priorità molto sofisticato che permette una grande elasticità nell'uso delle interruzioni vettoriali con un microcomputer 8080A.

Nello scrivere il software, dovete decidere a quali dispositivi è permessa l'interruzione e a quali no. Viene sviluppata una configurazione di bit di maschera, in cui ai dispositivi a cui è permessa l'interruzione è assegnato un livello logico 1 e a quelli a cui non è permessa, un livello logico 0. La configurazione di maschera a 8 bit è posta nell'accumulatore e messa in uscita sui due latch 7475 in Figura 23.15.

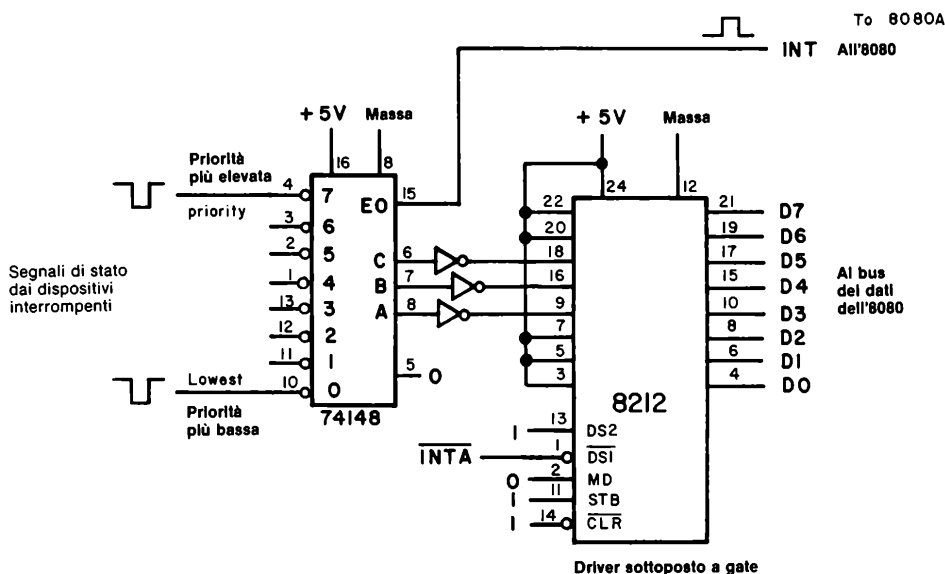



*Figura 23-14.* Circuito di interruzione con priorità da hardware che genera otto diverse istruzioni di ripristino vettorizzate, RST X, che hanno la priorità 7>6>5>4>3>2>1>0.

La posizione del bit D7 corrisponde al dispositivo di interrupt 7, che ha la massima priorità e dà luogo ad una vettorizzazione all'indirizzo di memoria HI = 000 e LO = 070. I dispositivi attivi che vengono mascherati, usano un ingresso del registro di lettura per richiedere il servizio; la maschera può essere cambiata sotto il controllo del software per raggiungere una notevole elasticità nell'uso delle interruzioni. Nella Figura 23-15, le richieste di interruzione sono poste in OR e le richieste di interruzione non mascherate vengono fatte arrivare al latch 74100. Quando l'uscita INTE del chip 8080A, di abilitazione all'interruzione, indica che le interruzioni saranno accettate, il 74100 è "aperto" e fa passare le richieste di interruzione fino al codificatore di priorità. Il codificatore di priorità e la porta di ingresso della istruzione d'interruzione sono stati trattati in precedenza. Quando viene ricevuta l'interruzione, l'8080A disabilita il suo flip-flop interno di abilitazione all'interruzione e l'uscita INTE va a livello logico 0, "chiudendo" così il latch 74100 ed eseguendo un latch su qualunque richiesta di interruzione presente agli ingressi. Il segnale di controllo $\overline{\text{INTA}}$ non solo fa accettare l'istruzione RST X, ma manda anche gli opportuni impulsi al decodificatore 7442 della Fig. 23-15, al fine di azzerare il flag richiesta interrupt del dispositivo interrompente. Si possono usare molti altri schemi di interruzione ad esempio utilizzare l'interrupt controller 8214 della Intel e il chip di controllo interruzione programmabile 8259. Infine, ricordatevi che le interruzioni, se permettono risposte veloci a domande esterne di servizio, presentano però anche dei problemi.

*Figura 23-15.* Un esempio di sofisticato circuito di interruzione con priorità, che genera impulsi individuali di azzeramento dei flag ed ha un registro maschera in modo che i dispositivi possono essere mascherati per mezzo dell'hardware esterno.

## IL SOFTWARE DI GESTIONE DELLE INTERRUZIONI

Consideriamo ora il software necessario per servire alcune delle nostre necessità di interruzione. Supponiamo di avere sole due dispositivi, il dispositivo 7, che ha la massima priorità, e un dispositivo a priorità, minore, il dispositivo 2. Ognuno di essi ha la sua istruzione di ripristino che dà luogo ad un vettore a 000 070 o 000 020, rispettivamente. Più avanti supporremo che il dispositivo a priorità maggiore interrompe su una base regolare e viene servito velocemente con la sua routine di servizio dell'interruzione. Il dispositivo 2, a priorità minore, interrompe in modo irregolare ed occupa del tempo per il suo servizio. Forse il dispositivo 2 è un altro microcomputer che sta riversando blocchi di dati. Quando non è interrotto da questi dispositivi, il microcomputer eseguirà sempre il software MAIN TASK. Infine, supporremo che il MAIN TASK inizialmente assegna uno stack pointer attraverso l'istruzione LXI SP<B2><B3> ed abilita anche il flag di interrupt.

Dato che le nostre interruzioni possono avvenire in qualunque momento, abbiamo bisogno sia dell'istruzione PUSH che dell'istruzione POP nelle routine di servizio dell'interruzione, di cui abbiamo già dato un esempio in Figura 23-10. Queste istruzioni conserveranno e memorizzeranno di nuovo qualunque registro venga alterato nelle routine di servizio.

L'esecuzione del software può essere rappresentata graficamente da una sequenza temporale, come mostra la Figura 23-16. Notate che il dispositivo a priorità superiore ha interrotto il MAIN TASK quattro volte e che il dispositivo a priorità minore l'ha interrotto solo una volta. Il dispositivo a priorità maggiore interrompe in modo regolare, come mostrano gli spazi sulla sequenza del MAIN TASK. La linea in grassetto indica quando l'interruzione è abilitata. La sequenza temporale mostrata è inesatta, dato che si vede solo il tempo trascorso nel MAIN TASK. È più corretto far vedere il tempo reale trascorso sia nel MAIN TASK che nelle subroutine, come abbiamo fatto nella Figura 23-17.

Nella Figura 23-17 il MAIN TASK iniza ad operare ed è poi interrotto dal dispositivo a priorità superiore. Dopo aver eseguito la subroutine di servizio del dispositivo a priorità superiore, il controllo è ridato al MAIN TASK, che è poi interrotto a intervalli ripetuti dal dispositivo a priorità superiore. *Notate che occorre molto più tempo a raggiungere il punto # nel MAIN TASK, quando continuiamo a interromperlo.* In momenti critici, ciò potrebbe essere disastroso, ad esempio se stiamo realizzando dei loop di temporizzazione.

Dato che il dispositivo a priorità superiore interrompe in modo regolare, esso ha probabilmente tentato di interrompere nel tempo in cui il microcomputer stava lavorando il software di servizio per il dispositivo a priorità inferiore. Perchè il dispositivo a priorità superiore non può interrompere il software del dispositivo a priorità inferiore? La risposta è ovvia: *il flag di interrupt non era abilitato durante l'esecuzione della subroutine di servizio del dispositivo a priorità inferiore.* Il nostro primo tentativo di scrivere il software di servizio della interruzione non prendeva in considerazione questa possibilità. I dati o i segnali del dispositivo a priorità superiore venivano persi nel tempo. Per risolvere questo problema, possiamo correggere il nostro software mettendo l'istruzione di abilitazione all'interruzione all'inizio della subroutine di servizio interruzione del dispositivo di priorità inferiore anzichè alla fine. Possiamo anche progettare dell'hardware per memorizzare i dati o i segnali associati all'interruzione persa.

Spostando l'istruzione di abilitazione all'interruzione, EI, verso l'inizio della subroutine di servizo di priorità piu bassa, possiamo incontrare un nuovo problema: il flusso del software del dispositivo a priorità inferiore è interrotto, come mostra la Figura 23-18. Per mettere in rilievo il fatto, abbiamo supposto che il dispositivo a priorità superiore interrompa il dispositivo a priorità inferiore due volte, dividendo così il software inferiore in tre pezzi.

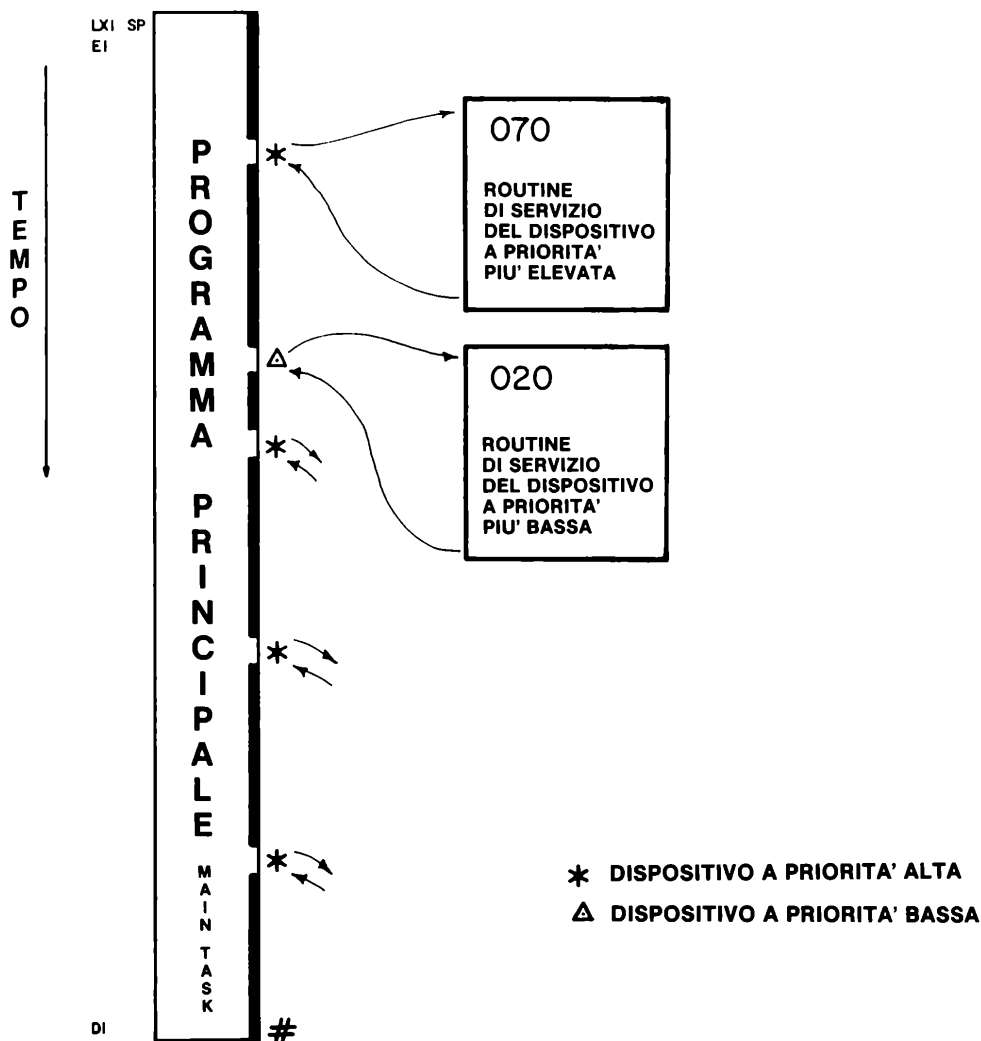



*Figura 23-16.* Sequenza temporale della esecuzione del programma per il MAIN TASK. Le interruzioni dei dispositivi a priorità superiore e inferiore sono indicate dai simpoli, * e △, rispettivamente.

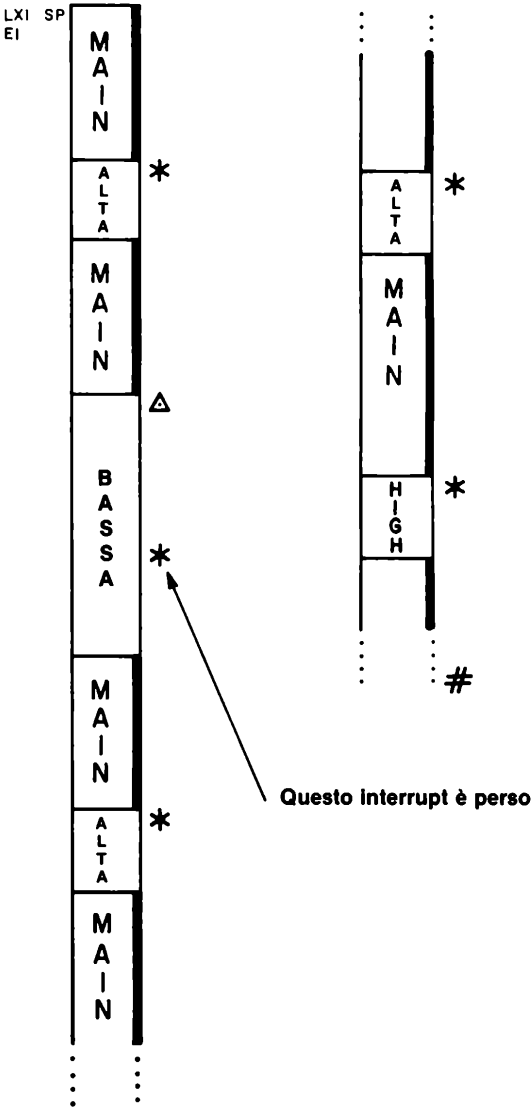



*Figura 23-17.* Sequenza temporale della esecuzione del programma per il MAIN TASK e le subroutine di servizio dei dispositivi di priorità superiore e inferiore.

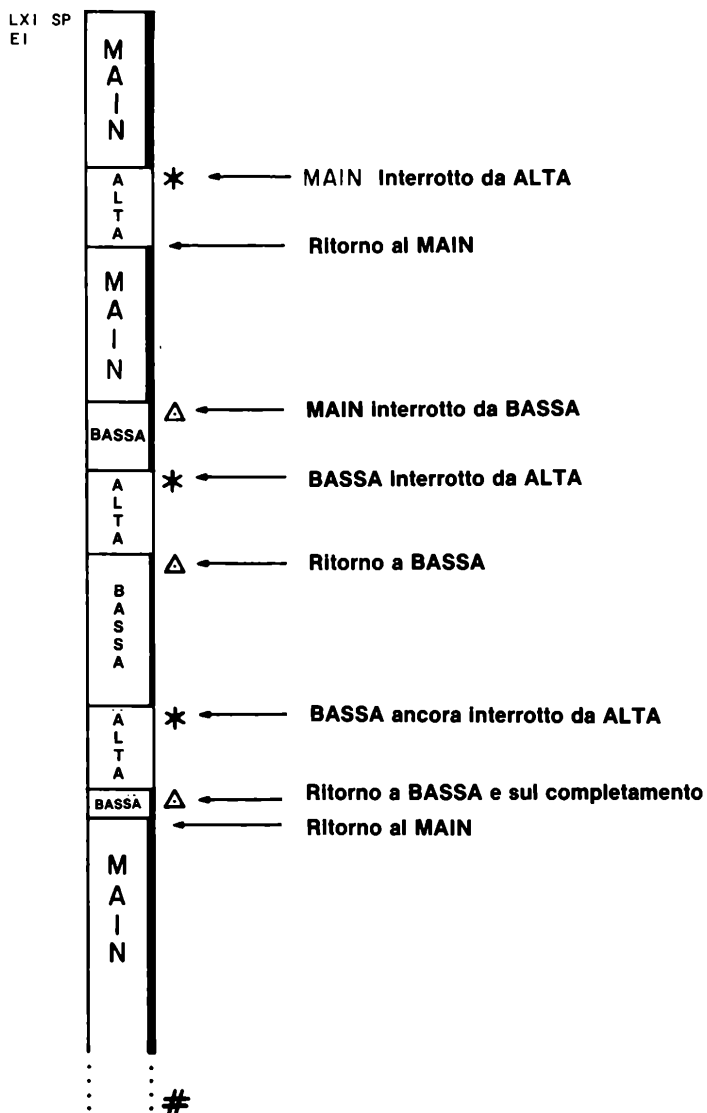



*Figura 23-18.* Sequenza temporale della esecuzione del programma che mostra la routine di servizio dell'interruzione. Il software di interruzione inferiore è interrotto due volte dal dispositivo a priorità superiore.

Con il software del dispositivo a priorità inferiore così suddiviso, dobbiamo stabilire se siamo in grado di completare questo software *prima che il dispositivo a priorità inferiore generi una nuova interruzione.* È assolutamente possibile per il dispositivo a priorità inferiore interrompere il microcomputer mentre esso sta ancora tentando di servire l'ultima richiesta di interruzione del dispositivo a priorità inferiore. Mentre la risposta dell'interruzione è veloce, l'effettivo tempo di esecuzione può essere molto più lungo del tempo richiesto per un solo passo attraverso il software del servizio interruzione. Questo accade perchè possiamo interrompere le nostre interruzioni. È molto semplice per un microcomputer diventare legato alle interruzioni, cioè esso passa tutto il suo tempo a provare e a servire le interruzioni e non gli resta più tempo per il software del MAIN TASK.

Nel nostro software, possiamo voler evitare che le interruzioni abbiamo luogo per la presenza di software sensibile al tempo o di compiti o calcoli complessi dipendenti dal tempo. L'istruzione di disabilitazione all'interruzione permette al computer di operare a tali condizioni, insensibile ad interruzioni esterne. Possiamo sempre disabilitare il flag di interrupt e riabilitarlo più tardi quando abbiamo completato un compito sensibile. Comunque, nell'intervallo di tempo in cui il flag di interrupt è disabilitato, *possiamo perdere segnali o dati di cui un dispositivo di interruzione può aver bisogno per inserirli nel calcolatore.* Tale situazione è rappresentata nella Figura 23-19. A meno che non forniamo dei tipi complessi di hardware d'emergenza, tali dati sono perduti! Non sappiamo esattamente quando un dispositivo esterno può interrompere MAIN TASK. Perciò, non possiamo essere sicuri che tale interruzione non sia nel momento in cui il flag di interrupt è disabilitato. Come possiamo aggirare il problema? Non è facile da fare, e questa è un'altra ragione per cui dobbiamo fare grande attenzione nell'usare le interruzioni.

Vi è un altro tipo di interruzione che può essere interessante, sebbene non venga usata generalmente con un microcomputer 8080A: è l'interruzione *"orientata nel tempo."* Si usa solo un'interruzione, un clock. Il clock interrompe ogni 10 millisecondi, o un altro periodo di tempo ragionevole. Quando viene interrotto, il microcomputer usa una tabella per determinare quali dispositivi deve provare a vedere se hanno bisogno di servizio. Alcuni dispositivi vengono provati sempre, mentre altri dispositivi più lenti possono essere provati una volta su cento interruzioni del clock. Questa è una buona tecnica di interruzione alternativa, ma richiede una notevole quantità di software per opere bene.

I tipi di microprocessori 8080A più nuovi permettono ad istruzioni a più byte di essere inserite durante un'interruzione, da cui l'utilizzo di istruzioni a 3 byte o di un salto completo, eliminando così le posizioni dei vettori e fornendo una maggiore flessibilità sia nell'hardware che nel software. La chiave per le istruzioni a più byte è il controllore 8228, che ha la capacità di generare segnali di controllo $\overline{\text{INTA}}$ in successione, in risposta alla richiesta d'interruzione. Questi tre segnali sono usati dall'hardware per fornire successivamente i tre byte di un'istruzione di richiamo o di salto. Il controllore d'interruzione programmabile 8259 opera assieme al chip 8228 per permettere di eseguire direttamente richiami alle subroutine di servizio dell'interruzione. Se il vostro microcomputer basato sull'8080A non contiene un 8228, non sarete in grado di usare l'8259, che è un dispositivo complesso, di non facile utilizzo per principianti.

Alcune precauzioni per finire. Le interruzioni sono difficili da mettere a punto. Possono avvenire quasi in qualunque momento; cioè avvengono in modo asincrono. I programmi tipici di messa a punto del software non sono di molto aiuto. Può essere necessario scrivere uno speciale software diagnostico per controllare le interruzioni in un'applicazione specifica. Quando prendete in considerazione le interruzioni, tentate tutti gli altri metodi prima di fermarvi su di esse. Il tempo che impiegherete sarà in genere ben speso.

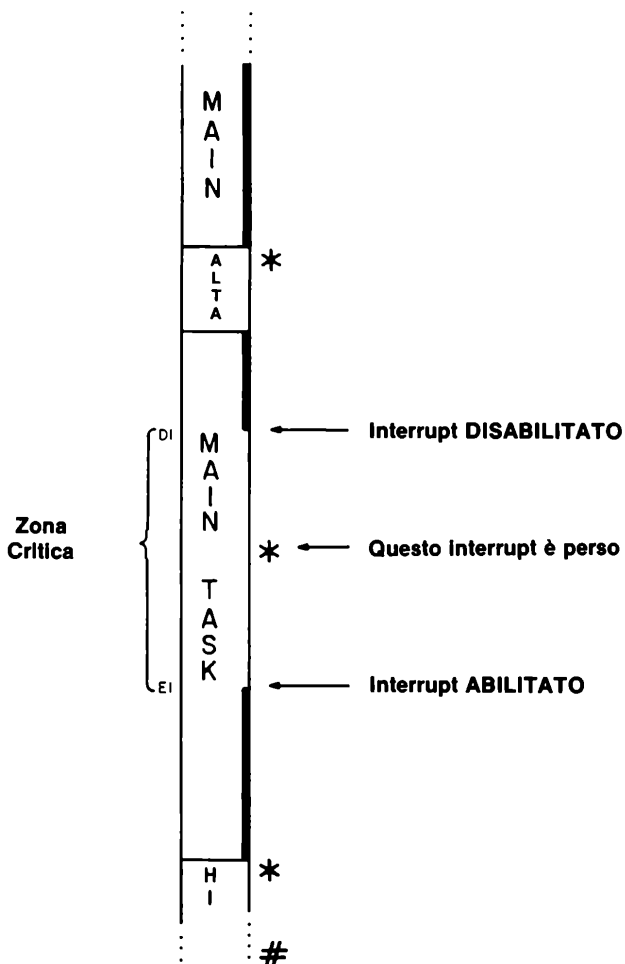



*Figura 23-19.* Sequenza temporale della esecuzione di un programma che dimostra l'uso di istruzioni DI ed EI per far sì che venga eseguito un compito critico nel MAIN TASK. In questo caso, comunque, si perde o si ritarda un'interruzione mentre il compito viene eseguito. È del tutto possibile che vengano persi dei dati per la mancata interruzione a meno che non venga fornito hardware esterno per una tale situazione.

## INTRODUZIONE AGLI ESPERIMENTI

Gli esperimenti seguenti illustrano l'uso dei flag e delle interruzioni.

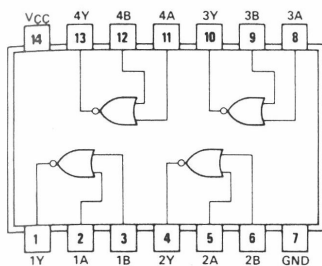
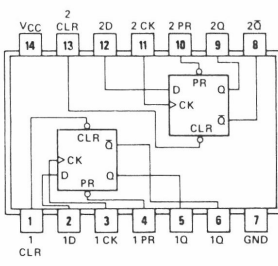
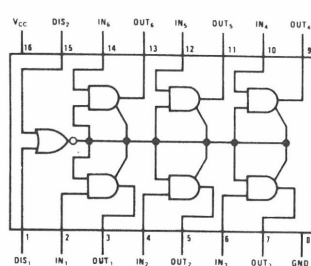
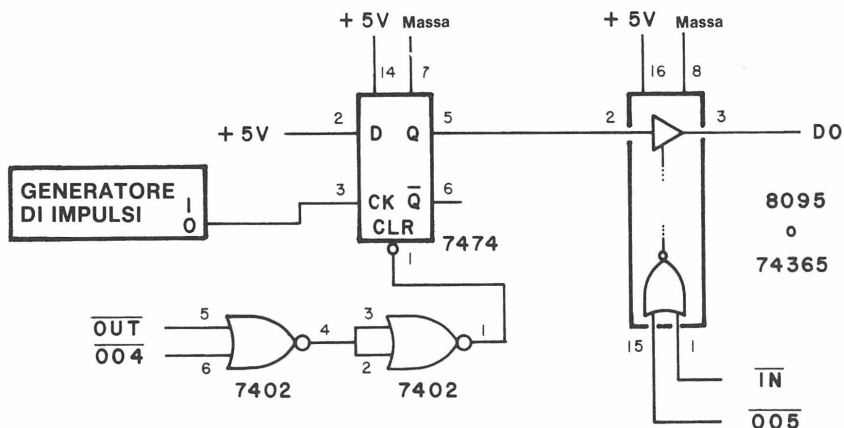
| Esperimento N. | Commento |
|---|---|
| 1 | Un semplice flag. Dimostra l'operazione di un semplice circuito di flag esterno, costituito da un flip-flop 7474 e da un buffer three-state 8095 (74365). |
| 2 | Tempo di risposta del flag. Illustra la risposta del software ai flag quando il microcomputer ha altri compiti da eseguire. |
| 3 | Flag non ideali: l'interfacciamento di uno switch meccanico. Illustra l'operazione di un circuito di flag esterno che è collegato ad un switch ad un solo polo (SPST), dispositivo meccanico non ideale. |
| 4 | Caratteristiche della tastiera del microcomputer MMD-1. Dimostra come usare il flag della tastiera, il bit D7, per segnalare che è stato premuto un tasto e i dati sono pronti per essere inseriti nel micocomputer. |
| 5 | Simulazione della lettura del livello di un liquido in un serbatoio. Implementa l'hardware e il software necessario per simulare l'esempio di lettura del livello di un liquido, discusso nel testo. |
| 6 | Istruzioni di ripristino. Illusta le caratteristiche software delle istruzioni di ripristino dell'8080A, RST X. |
| 7 | Un semplice registro di istruzione d'interruzione. Illustra il comportamento di un registro di istruzione ricavato da un buffer/latch 8212. |
| 8 | Come forzare un'istruzione di ripristino. Dimostra le conseguenze dell'inserimento di un'istruzione di ripristino nel registro di istruzioni 8212, cablato nell'Esperimento N. 7. |
| 9 | Tempo di risposta ad una interruzione. Illustra la risposta di un sistema basato sull'8080 alle interruzioni, quando il microcomputer ha altri compiti da eseguire. |
| 10 | Semplici interruzioni con priorità. Illustra l'implementazione di un semplice schema di interruzione con priorità, che comprende un dispositivo a priorità inferiore ed uno a priorità superiore. |
| 11 | Calcolo dei tempi relativi ad una interruzione con priorità. Illustra la relazione temporale fra i dispositivi a priorità superiore e inferiore e come viene assegnata la priorità. |
| 12 | Interruzioni simultanee. Illustra le operazioni di interruzione simultanea. |

## ESPERIMENTO N. 1

## UN SEMPLICE FLAG

### Scopo

Lo scopo di questo esperimento è dimostrare il comportamento di un semplice flag esterno.

### Configurazioni dei pin dei circuiti integrati

### Schema del circuito

**Programma**

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | 227 | SUB A | /Azzera il registro A (accumulatore) |
| 003 001 | 107 | MOV B,A | /Sposta A verso B |
| 003 002 | 323 | OUT | /Poni in uscita i contenuti di A |
| 003 003 | 002 | 002 | /sulla porta di uscita 002 |
| 003 004 | 333 | INPUT, IN | /Inserisci i dati in A |
| 003 005 | 005 | 005 | /dalla porta d'ingresso 005 |
| 003 006 | 037 | RAR | /Rotazione del bit D0 nel flag di carry |
| 003 007 | 322 | JNC | /CARRY è = 0? Se sì, salta a |
| 003 010 | 004 | INPUT | /HI = 003 e LO = 004. Se no, |
| 003 011 | 003 | Ø | /passa all'istruzione successiva |
| 003 012 | 323 | OUT | /No, perciò poni in uscita un impulso |
| 003 013 | 004 | 004 | /di selezione dispositivo sulla porta 004 |
| 003 014 | 170 | MOV A,B | /Sposta B verso A |
| 003 015 | 074 | INR A | /Incrementa A |
| 003 016 | 107 | MOV B,A | /Sposta A verso B |
| 003 017 | 323 | OUT | /Poni in uscita i contenuti di A |
| 003 020 | 002 | 002 | /sulla porta di uscita 002 |
| 003 021 | 303 | JMP | /Salta indietro a INPUT a |
| 003 022 | 004 | FLAG | /LO = 004 e |
| 003 023 | 003 | Ø | /HI = 003 |

**Passo 1**

Montate il circuito mostrato. Assicuratevi che i collegamenti a massa e + 5 V siano fatti per tutti i chip e per i bus di alimentazione sulla piastra SK-10.

**Passo 2**

Caricate il programma nella memoria di lettura/scrittura iniziando a HI = 003 e LO = 000. Osservate che il formato del programma è diverso da quello usato nel testo in questo Capitolo nonchè in quelli precedenti. Questo è il tipo di uscita che dovreste ottenere da un assembler residente commerciale come quello messo a disposizione dalla Tychon, Inc. Notate l'uso del *delimitatore, /*, che è un carattere che indica l'inizio di un commento.

Il programma inserirà il bit di flag che abbiamo cablato, testerà il bit di flag D0 per determinare se è a livello logico 1, e poi incrementerà il registro A e metterà in uscita i dati sulla porta di uscita 002.

**Passo 3**

Eseguire il programma. Che cosa osservate alla porta di uscita 000?

La lettura della porta di uscita è 000, cioè tutti gli otto LED sono spenti.

**Passo 4**

Ora premete e rilasciate il generatore di impulsi. Che cosa cambia alla porta di uscita 002? Premete e rilasciate ripetutamente. Che cosa cambia?

Il primo impulso di clock del generatore di impulsi incrementa la porta di uscita 000 di 1 in modo che la lettura diventa 001. Gli impulsi clock aggiuntivi continuano ad incrementare la porta di uscita.

**Passo 5**

Se fate un errore nel cablaggio del circuito e cablate l'uscita $\overline{Q}$ (pin 16) del flip-flop 7474 del chip 74365, il programma opererà correttamente? Cambiate il collegamento dell'uscita 7474 con $\overline{Q}$ al pin 6 ed eseguite ancora una volta il programma. Che cosa osservate? Perchè?

Con l'ingresso $\overline{Q}$ verso il microcomputer, abbiamo osservato che tutti gli indicatori a LED sulla porta di uscita 002 sembravano accese. La ragione è che il programma ha rivelato che il bit D0 era continuamente a livello logico 1. A causa della natura del programma, i contenuti del registro A erano incrementati continuamente e messi in uscita sulla porta 000.

**Passo 6**

Potreste cambiare il software per tener conto degli errori nel montaggio? Se sì, che cambiamenti fate? Fate tali cambiamenti ed eseguire di nuovo il programma. Che cosa osservate?

Per eliminare l'effetto dell'errore nel montaggio, l'istruzione JNC a 003 007 può essere cambiata in un'istruzione JC, 332. Fatto questo cambiamento, il circuito e il programma non si comportano come descritto nel passo 4.

*Riportate l'hardware e il software alle loro forme originali e passate all'esperimento successivo.*

## ESPERIMENTO N. 2

## TEMPO DI RISPOSTA DI UN FLAG

### Scopo

Lo scopo di questo esperimento è controllare la risposta del software ai flag quando il microcomputer ha altri compiti da eseguire.

### Schema del circuito

Il circuito è identico a quello dato nell'Esperimento N. 1

### Programma

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | | | |
| . | | | |
| . | *Questi passi di programma sono gli stessi dell'esperimento N. 1.* | | |
| . | | . | |
| 003 021 | 016 | MVI C | /Carica il registro C con il |
| 003 022 | 001 | 001 | /byte di dati 001 |
| 003 023 | 315 | REPEAT, CALL | /Richiama la subroutine a ritardo di |
| 003 024 | 277 | TIMEOUT | /tempo KEX, TIMEOUT a LO = 277 e |
| 003 025 | 000 | Ø | /HI = 000 |
| 003 026 | 015 | DCR C | /Decrementa il registro C |
| 003 027 | 302 | JNZ | /Il registro C è = 000? Se no, salta a |
| 003 030 | 023 | REPEAT | /REPEAT a LO = 023 e |
| 003 031 | 003 | Ø | /HI = 003. Altrimenti continua. |
| 003 032 | 303 | JMP | /Si, il registro C = 000. Salta indietro a |
| 003 033 | 004 | INPUT | /INPUT a LO = 004 e |
| 003 034 | 003 | Ø | /HI = 003 |

### Passo 1

In questo esperimento si useranno l'hardware e il software di quello precedente. Sono stati aggiunti passi di software in più per tenere il microcomputer occupato per la variazione dei periodi di tempo. Viene usata la subroutine a ritardo di tempo KEX, di 10 millisecondi a 000 277. Un listing di TIMEOUT è fornito al termine di questo esperimento.

In questo esperimento, usiamo l'area di stack KEX per conservare l'indirizzo di ritorno per l'istruzione CALL. Se non usate KEX su di un microcomputer MMD-1, dovrete prima stabilire un'area di stack. Usate l'istruzione LXI SP per farlo.

Caricate il programma suddetto nella memoria di lettura/scrittura iniziando a 003 021.

**Passo 2**

Iniziate l'esecuzione del programma a 003 000. Premete e rilasciate il generatore di impulsi parecchie volte. Osservate una differenza fra questo esperimento e il precedente, dove in conteggio aumentava per ogni impulso clock del generatore di impulsi?

Non abbiamo osservato nessuna differenza.

**Passo 3**

Il software aggiunto rallenta il microcomputer fornendo una routine a ritardo di tempo di 10 millisecondi da eseguire. Posizionando il registro C su un altro valore, farete sì che il microcomputer esegua la routine a ritardo di tempo molte più volte, rallentando così ancora di più tutto il loop.

Inserite i seguenti byte di timing, uno alla volta nel programma, alla locazione di memoria 003 022 ed eseguite il programma in tutti i casi. Controllare l'influenza di ogni byte (a) fornendo con lentezza parecchi impulsi di clock dal generatore di impulsi, e (b) fornendo dieci impulsi di clock dal generatore più in fretta che potete. Inserite il numero di conteggi che osservate nella tabella seguente.

| Byte ottale di timing | Ritardo di tempo | Normale | Dieci impulsi veloci |
|---|---|---|---|
| 012 | 100ms | | |
| 024 | 200ms | | |
| 062 | 500ms | | |
| 144 | 1s | | |
| 310 | 2s | | |

Quando abbiamo fornito lentamente gli impulsi di clock al flag, abbiamo osservato un comportamento normale, cioè l'uscita della porta aumentava una volta per ogni impulso di clock. Comunque, quando abbiamo applicato dieci impulsi veloci, abbiamo osservato solo cinque impulsi conteggiati con il ritardo di tempo di 500 ms, tre impulsi conteggiati con il ritardo di 1 secondo e solo due impulsi conteggiati con il ritardo di 2 secondi.

**Passo 4**

Che cosa indica il risultato nel passo 3 circa l'uso dei flag quando il microcomputer ha da eseguire altri compiti che occupano tempo?

Eventi o dati possono andar perduti dato che non vengono letti dal microcomputer quando esso sta eseguendo altri compiti.

*Conservate il vostro hardware e il vostro software per l'esperimento successivo.*

**LISTING DELLA SUBROUTINE TIMEOUT**

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 000 277 | 365 | TIMEOUT, PUSHPSW | /Salva l'accumulatore e i flag |
| 000 300 | 325 | PUSHD | /Salva la coppia di registri D |
| 000 301 | 021 | LXI D | /Carica D ed E con i valori che |
| 000 302 | 046 | 046 | /vanno decrementati |
| 000 303 | 001 | 000 | |
| 000 304 | 033 | MORE, DCX D | /Decrementa la coppia di registri D |
| 000 305 | 172 | MOV A,D | /Sposta D verso A |
| 000 306 | 263 | ORA E | /OR di E con A |
| 000 307 | 302 | JNZ | /Il registro A è = 000? Se no, salta |
| 000 310 | 304 | MORE | /a MORE a LO = 304 e |
| 000 311 | 000 | Ø | /HI = 000. Altrimenti passare all'istruzione seguente |
| 000 312 | 321 | POP D | /Memorizza di nuovo la coppia di /registri D |
| 000 313 | 361 | POP PSW | /Memorizzare di nuovo l'accumulatore /e i flag |
| 000 314 | 311 | RET | /Rientro dalla subroutine TIMEOUT |

Con un microcomputer 8080A che opera a 750 kHz, questa routine a ritardo di tempo genererà un ritardo di 10 millisecondi.

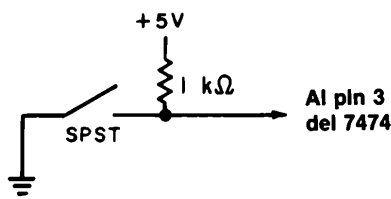
## ESPERIMENTO N. 3

## FLAG NON IDEALI: L'INTERFACCIAMENTO CON UNO SWITCH MECCANICO

**Scopo**

Lo scopo di questo esperimento è controllare il funzionamento di un circuito di flag esterno collegato ad un switch a un polo (SPST): un dispositivo meccanico non ideale.

**Schema del circuito**

Il circuito è identico a quello dato nell'Esperimento N. 1 eccetto che per l'ingresso al flip-flop 7474 del generatore d'impulsi. Al posto di quest'ultimo, usate un switch meccanico a un polo (SPST) cablato nel modo seguente:

**Programma**

Il programma è identico a quello dell'Esperimento N. 1. Assicuratevi che l'istruzione di salto finale si presenti nel seguente modo:

| | | | |
|---|---|---|---|
| 003 021 | 303 | JMP | /Salta indietro a INPUT a |
| 003 022 | 004 | INPUT | /LO = 004 e |
| 003 023 | 003 | Ø | /HI = 003 |

**Passo 1**

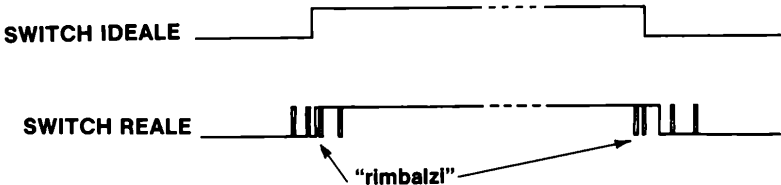
Anche in questo esperimento si useranno l'hardware e il software degli esperimenti precedenti, Se non sono ancora pronti, cablate il circuito mostrato nell'Esperimento N. 1. Al posto del generatore d'impulsi, cablate lo switch SPST o ancora usate un switch logico presente sull'Outboard LR-2 o LR-25.

**Passo 2**

Il software usato in questo esperimento è identico a quello dell'Esperimento N. 1. Assicuratevi che venga caricato in modo corretto nella memoria di lettura /scrittura.

**Passo 2**

Eseguite il programma ed utilizzato lo switch meccanico SPST non ideale (o un circuito equivalente). Esaminate l'uscita alla porta di uscita 000. Osservate un solo conteggio, come nell'Esperimento N. 1, o molti conteggi? Perchè?

Abbiamo osservato molti conteggi. Questo indica che il nostro switch SPST non è uno switch "ideale", come un generatore di impulsi "senza rimbalzi". La differenza fra i due switch si può raffigurare come segue:

Abbiamo osservato molti conteggi perchè il flag 7474 ha letto ogni "rimbalzo" come una chiusura individuale dello switch.

**Passo 4**

Azionate lo switch SPST dieci volte ed osservate il numero totale dei conteggi. Ripetete questo processo parecchie volte, assicurandovi di indirizzare il microcomputer prima di ogni tentativo. Riassumete i vostri risultati nella tabella seguente accanto ai nostri risultati. Notate che il numero di conteggi è espresso in codice ottale e non decimale.

| Tentativo | Nostri risultati | Vostri risultati |
|---|---|---|
| 1 | 62 | |
| 2 | 36 | |
| 3 | 67 | |
| 4 | 70 | |
| 5 | 160 | |

Il loop del microcomputer non era abbastanza veloce per rivelare tutti i rimbalzi del vostro switch meccanico imperfetto, e i rimbalzi non sono riproducibili. Mentre i rimbalzi possono essere eliminati con l'hardware, come abbiamo fatto nel primo esperimento, possono essere eliminati anche usando il software. L'esperimento che segue esamina l'interfaccia per una tastiera e dimostra come usare il software KEX per "filtrare" i rimbalzi.

## ESPERIMENTO N. 4

## CARATTERISTICHE DELLA TASTIERA DEL MICROCOMPUTER MMD-1

### Scopo

Lo scopo di questo esperimento è dimostrare come si usa il flag della tastiera, il bit D7, per segnalare che un tasto è stato premuto e i dati sono pronti per essere inseriti nel microcomputer MMD-1.

### Programma N. 1

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 100 | 333 | INPUT, IN | /Inserisci i dati della tastiera dalla |
| 003 101 | 000 | 000 | /porta d'ingresso 000 |
| 003 102 | 027 | RAL | /Rotazione del bit D7 nel flag di carry |
| 003 103 | 322 | JNC | /Se CARRY è a livello logico 0, salta a |
| 003 104 | 100 | INPUT | /INPUT a LO = 100 e HI = 003. |
| 003 105 | 003 | Ø | /Altrimenti, passa all'istruzione /successiva. |
| 003 106 | 037 | RAR | /Se CARRY è a livello logico 1, fai /ruotare indietro i dati |
| 003 107 | 323 | OUT | /Poni in uscita i dati sulla |
| 003 110 | 002 | 002 | /porta di uscita 002 |
| 003 111 | 303 | JMP | /Salta indietro a INPUT a |
| 003 112 | 100 | INPUT | /LO = 100 e |
| 003 113 | 003 | Ø | /HI = 003 e ripetere |

### Programma N. 2

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 200 | 315 | START, CALL | /Richiama la subroutine della tastiera |
| 003 201 | 315 | KBRD | /KBRD a LO = 315 e |
| 003 202 | 000 | Ø | /HI = 000 |
| 003 203 | 323 | OUT | /Poni in uscita i dati della tastiera sulla |
| 003 204 | 002 | 002 | /porta di uscita 002 |
| 003 205 | 303 | JMP | /Salta indietro a START a |
| 003 206 | 200 | START | /LO = 200 e |
| 003 207 | 003 | Ø | /HI = 003 e ripetere |

**Passo 1**

Questo esperimento non richiede un circuito interno di interfaccia. Usate la tastiera MMD-1 per generare il bit di flag e i dati della tastiera. Caricate il programma N. 1 nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 100. Quale bit userete nell'accumulatore per segnalare al chip 8080A che è stato premuto un tasto?

Userete il flag della tastiera, il bit D7, per segnalare all'8080A che è stato premuto un tasto e che i dati sono pronti per essere inseriti.

**Passo 2**

Eseguite il programma ed esaminate i quattro bit meno significativi alla porta di uscita 002. Premete a turno tutti i tasti della tastiera e fate un elenco del codice a 4 bit che osservate sotto la scritta "Passo 2 nella tabella seguente". Osservate un'equivalenza fra il vostro codice e quello che vi aspettavate?

| Tasto | Codice previsto | Il Vostro codice | |
|---|---|---|---|
| | | Passo 2 | Passo 3 |
| 0 | 0000 | | |
| 1 | 0001 | | |
| 2 | 0010 | | |
| 3 | 0011 | | |
| 4 | 0100 | | |
| 5 | 0101 | | |
| 6 | 0110 | | |
| 7 | 0111 | | |
| H | 1100 | | |
| L | 1101 | | |
| G | 1011 | | |
| S | 1000 | | |
| A | 1110 | | |
| B | 1111 | | |
| C | 1010 | | |

Abbiamo osservato un'equivalenza fra il nostro codice per il passo 2 ed il codice previsto.

**Passo 3**

Se i vostri codici non equivalgono al nostro codice previsto, potete usare la subroutine di "filtro" KEX, documentata nel Manuale dell'MMD-1. Il programma N. 2. richiama la subroutine d'ingresso della tastiera KBRD in KEX. Caricate il programma N. 2 nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 200 ed eseguitelo. Premete tutti i tasti della tastiera a turno e notare i bit meno significativi alla porta di uscita 002. Elencate questi quattro bit sotto la colonna Passo 3 nella tabella suddetta. I codici per Passo 2 e Passo 3 sono gli stessi? Se no, perchè?

I codici per i tasti L,H,G,S,A,B e C sono stati cambiati. Perciò, non sono gli stessi.

**Passo 4**

Sapete suggerire perchè questa traslazione di codice dei tasti da L a C potrebbe essere utile?

Essa fornisce flessibilità e ci permette di ridefinire i tasti in software. KEX attua una tastiera decimale, se vogliamo usarla in quel modo. Notate che i tasti ora vanno da 0 a 11 in sequenza, saltando 12, e finiscono con 13, 14 e 15.

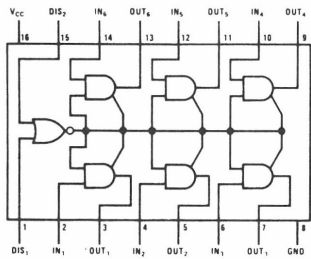
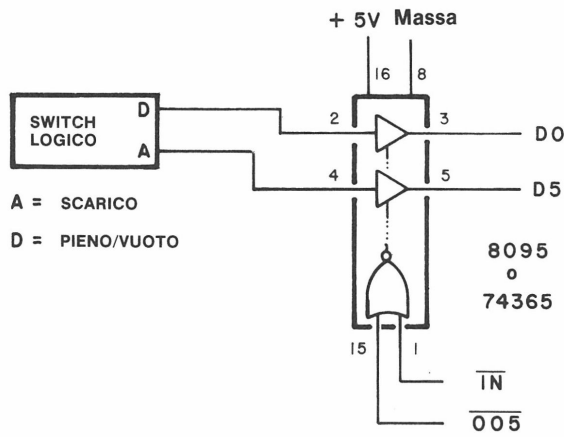
Con la KEX EPROM nel sistema, la routine d'ingresso della tastiera a 000 315 può essere chiamata per inserire e codificare i dati della tastiera. La routine della tastiera usa una subroutine a ritardo di 10 millisecondi a 000 277 che può anche essere richiamata in qualunque momento. La routine a ritardo è completamente "trasparente" e non influenzerà nessun flag o registro. Essa è elencata alla fine dell'Esperimento N. 2 di questo Capitolo.

## ESPERIMENTO N. 5

## SIMULAZIONE DELLA LETTURA DEL LIVELLO DI UN LIQUIDO IN UN SERBATOIO

### Scopo

Lo scopo di questo esperimento è implementare l'hardware e il software necessario per simulare l'esempio della lettura del livello di liquido discusso nel testo.

### Configurazione dei pin del circuito integrato

### Schema del circuito

**Programma**

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | 333 | START, IN | /Inserisci i dati dei flag dal |
| 003 001 | 005 | 005 | /dispositivo d'ingresso 005 |
| 003 002 | 346 | ANI | /Maschera tutti i bit eccetto il bit D5 |
| 003 003 | 040 | 040 | /Byte maschera = 00100000 |
| 003 004 | 302 | JNZ | /Se il risultato è 040, salta ad ALARM a |
| 003 005 | 100 | ALARM | /LO = 100 e HI = 003. Altrimenti, passa |
| 003 006 | 003 | Ø | /all'istruzione successiva |
| 003 007 | 333 | IN | /L'overflow è OK. Inserisci i dati di |
| 003 010 | 005 | 005 | /flag di nuovo dal dispositivo d'ingresso /005. |
| 003 011 | 017 | RRC | /Rotazione del bit D0 nel flag di carry |
| 003 012 | 332 | JC | /Se CARRY = 1, salta e FULL a |
| 003 013 | 024 | FULL | /LO = 024 e |
| 003 014 | 003 | Ø | /HI = 003. Altrimenti, passa /all'istruzione successiva |
| 003 015 | 076 | MVI A | /Se CARRY = 0, il serbatoio non è /pieno. |
| 003 016 | 001 | 001 | /Carica il registro A con il byte 001. |
| 003 017 | 323 | OUT | /Poni in uscita il byte sul |
| 003 020 | 000 | 000 | /dispositivo di uscita 000 |
| 003 021 | 303 | JMP | /Controlla l'overflow e i flag |
| 003 022 | 000 | START | /pieno/vuoto saltando ancora a |
| 003 023 | 003 | Ø | /LO = 000 e HI = 003 |
| 003 024 | 227 | FULL,SUB A | /Il serbatoio è pieno. Azzera il registro |
| 003 025 | 323 | OUT | /A. Poni in uscita il byte del registro A |
| 003 026 | 000 | 000 | /sul dispositivo di uscita 000 |
| 003 027 | 303 | JMP | /Controlla l'overflow ed i flag |
| 003 030 | 000 | START | /pieno/vuoto saltando ancora a |
| 003 031 | 003 | Ø | /START a LO = 000 e HI = 003 |
| 003 100 | 166 | ALARM, HLT | /Fine dell'operazione |

**Passo 1**

Montate il circuito mostrato nello schema. Lo switch logico A è il flag di overflow e lo switch logico D è il flag di pieno/vuoto.

**Passo 2**

Caricate il programma nella memoria di lettura/scrittura iniziando da HI = 003 e LO = 000.

**Passo 3**

Con entrambi gli switch logici settati sul livello logico 0, eseguire il programma. Simulerete la valvola con il bit D0 sulla porta di uscita 000. Se il bit D0 è a livello logico 1, la valvola è aperta; se il bit D0 è a livello logico 0, la valvola è chiusa.
Che cosa osservate quando inizia il programma? Perché?

Abbiamo visto che il bit D0 era a livello logico 1, indicando che la valvola era aperta. Questo succede perchè il flag di vuoto/pieno è a livello logico 0, indicando che il serbatoio non è pieno (o forse vuoto).

**Passo 4**

Cambiate lo switch logico D in livello logico 1, indicando così che il serbatoio è pieno. Che cosa succede al bit D0 della valvola? Perchè?

Il bit D0 della valvola a livello logico 0 passa, cioè indica che la valvola ora è chiusa. Il livello del liquido nel serbatoio ha modificato lo switch di pieno.

**Passo 5**

Commutate lo switch logico A a livello logico 1, indicando cioè una condizione di overflow. Che cosa succede? Perchè?

Non succede niente. Il microcomputer esegue un'istruzione di Halt alla locazione di memoria 003 100, la locazione della routine ALARM.

**Passo 6**

Sapete suggerire una routine ALARM adeguata al vostro microcomputer? Potete provare molte brevi routine ALARM. Usate lo spazio libero della pagina a lato.

Nello sviluppo di una routine ALARM, abbiamo deciso che la prima cosa da fare è chiudere la valvola e poi mettere in uscita una condizione di allarme alla porta 001. Il programma è il seguente:

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 100 | 227 | ALARM, SUB A | /Azzera il registro A |
| 003 101 | 323 | OUT | /Poni in uscita il byte del registro A |
| 003 102 | 000 | 000 | /sul dispositivo di uscita 000 |
| 003 103 | 057 | CMA | /Complementa il registro A |
| 003 104 | 323 | OUT | /Poni in uscita i contenuti del registro |
| 003 105 | 001 | 001 | /A sulla porta di uscita 001 e |
| 003 106 | 166 | HLT | /Arresta l'operazione |

Dovreste notare che in questo esperimento non avete usato i flip-flop come flag. Questa è una procedura valida nel caso specifico, dato che lo switch di overflow e lo switch pieno/vuoto manterrano i rispettivi stati fino a che non entriamo in azione. Un flip-flop di flag è generalmente usato in quei casi in cui il dispositivo che richiede attenzione da parte del microcomputer genera un breve impulso invece di un livello, linea cioè che diventa o alta o bassa per un tempo indefinito.

## ESPERIMENTO N. 6

## ISTRUZIONI DI RIPRISTINO

### Scopo

Lo scopo di questo esperimento è esaminare le caratteristiche software delle istruzioni di ripristino dell'8080A, RST X.

### Programma

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | 061 | LXI SP | /Carica lo stack pointer con il |
| 003 001 | 200 | 200 | /byte d'indirizzo LO e |
| 003 002 | 003 | 003 | /il byte d'indirizzo HI |
| 003 003 | 357 | RST 5 | /Richiama la subroutine a 000 050 |
| 003 004 | 166 | HLT | /Alt |
| 003 050 | 311 | RET | /Rientro dalla subroutine |

### Passo 1

Caricate il programma nella memoria di lettura/scrittura iniziando da 003 000. Se state eseguendo questo programma su un microcomputer MMD-1, ricordatevi quanto segue: *un'istruzione RST X richiama la subroutine a 000 0X0, ma il monitor KEX fa sì che il controllo di programma salti a 003 0X0, per le istruzioni di ripristino da RST 1 a RST 6. Le istruzioni di ripristino RST 0 e RST 7 sono usate dal monitor KEX.*
Potete avere conferma di questo fatto esaminando i contenuti delle locazioni di memoria EPROM a 000 010, 000 020, 000 030, 000 040, 000 050, e 000 060.

Quali byte di istruzione trovate nelle locazioni di memoria KEX da 000 050 a 000 052?

Troverete i seguenti byte di istruzioni:

| | | | |
|---|---|---|---|
| 000 050 | 303 | JMP | /Salta a |
| 000 051 | 050 | 050 | /LO = 050 e |
| 000 052 | 003 | 003 | /HI = 003 |

### Passo 2

Eseguite il programma. Osservate che lo stack pointer è settato all'indirizzo 003 200,

quindi lo stack inizierà ad una posizione di memoria di 1 inferiore a questa posizione di memoria, cioè 003 177. Dopo aver eseguito il programma, esaminate i contenuti delle due posizioni dello stack 003 176 e 003 177 e scrivete i contenuti di seguito.

| Posizione dello stack | Contenuti |
|---|---|
| 003 176 | |
| 003 177 | |

Questi due byte sono come quelli che vi aspettereste per l'esecuzione di un'istruzione CALL?

Abbiamo osservato nello stack quanto segue:

| Posizione dello stack | Contenuti |
|---|---|
| 003 176 | 004 |
| 003 177 | 003 |

I due byte dello stack corrispondono all'indirizzo di memoria, 003 004, che è l'indirizzo dell'istruzione HALT. Questo è esattamente quello che vi aspettereste per un richiamo. Dovreste notare che lo stack pointer punterà ancora sull'indirizzo di memoria 003 177 dopo l'esecuzione del programma. Perchè?

Non solo il programma ha eseguito un'istruzione RST 5, ma anche un'istruzione RET che ha estratto i due byte dallo stack. Sebbene estratti i valori originari sono rimasti nella memoria di lettura/scrittura, Ricordate che lo stack pointer è un registro interno dell'8080, e non può essere letto in maniera diretta.

**Passo 3**

La routine a 003 050 può eseguire anche altri compiti. Cambiate i passi del programma in:

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 050 | 170 | MOV A,B | /Sposta B verso A |
| 003 051 | 074 | INR | /Incrementa A |
| 003 052 | 107 | MOV B,A | /Sposta A indietro verso B |
| 003 053 | 323 | OUT | /Poni in uscita i contenuti di A sulla |
| 003 054 | 000 | 000 | /porta di uscita 000 |
| 003 055 | 311 | RET | /Rientro dalla subroutine |

Sul vostro microcomputer MMD-1, premete alternativamente i tasti RESET e G. Che cosa osservate? Perchè?

Abbiamo osservato che la porta di uscita 000 ha incrementato il suo conteggio per ogni ciclo RESET/G. Il software di incremento era richiamato dalla subroutine RST 5.

**Passo 4**

Dopo parecchi incrementi, esaminate di nuovo le posizioni dello stack 003 176 e 003 177. I valori dei byte sono diversi da quelli che avete osservato nel passo 2. Come spiegate questo risultato ricordando che avete eseguito la subroutine di ripristino parecchie volte?

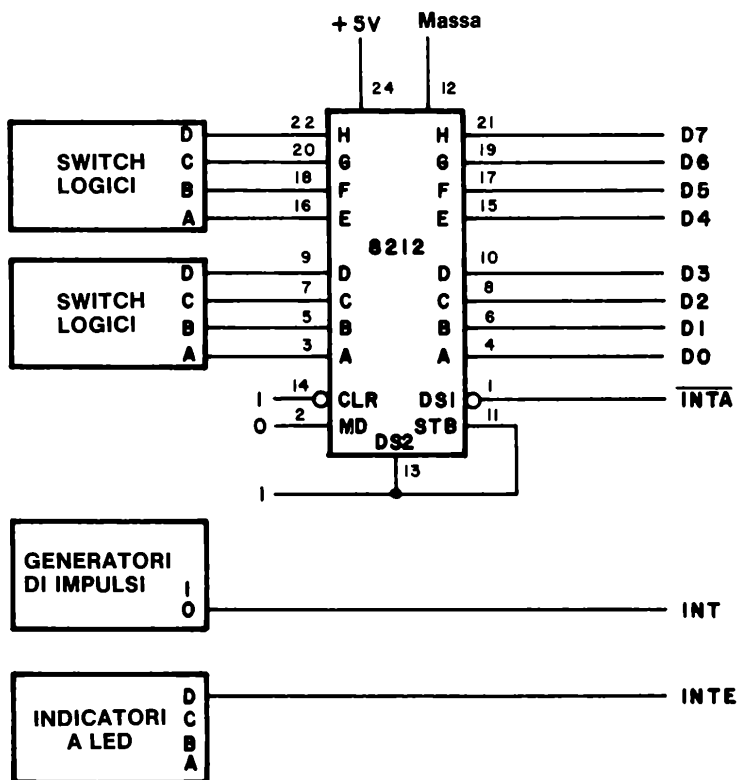
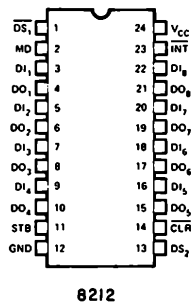
Non dovreste osservare nessun cambiamento nei byte contenuti nello stack, che sono ancora l'indirizzo dell'istruzione HALT, 003 004. Ogni volta che caricate un indirizzo sullo stack, come un risultato dell'istruzione RST 5, voi estraete anche gli stessi due byte quando eseguite l'istruzione RET. Chiaramente, per ogni ciclo RESET/G, e quindi per ogni richiamo della subroutine, l'indirizzo di rientro era lo stesso. Ricordate che l'indirizzo di rientro, memorizzato sullo stack, è per l'istruzione che segue l'istruzione CALL a uno o a tre byte. In questo caso, può darsi che sia un'istruzione HLT.

## ESPERIMENTO N. 7

## UN SEMPLICE REGISTRO DI ISTRUZIONE D'INTERRUZIONE

### Scopo

Lo scopo di questo esperimento è cablare e provare un semplice registro di istruzione di interruzione.

### Schema del circuito

**Configurazione dei pin del circuito integrato**

8212

**Programma**

| Indirizzo di memoria | Byte di istruzione | | Codice mnemonico | Commento |
|---|---|---|---|---|
| 003 000 | 061 | | LXI SP | /Carica lo stack pointer con |
| 003 001 | 200 | | 200 | /il byte d'indirizzo LO e |
| 003 002 | 003 | | 003 | /il byte d'indirizzo HI |
| 003 003 | 373 | | EI | /Abilita l'interruzione |
| 003 004 | 000 | REPEAT, | NOP | /Nessuna operazione |
| 003 005 | 000 | | NOP | /Nessuna operazione |
| 003 006 | 000 | | NOP | /Nessuna operazione |
| 003 007 | 170 | | MOV A,B | /Sposta B verso A |
| 003 010 | 323 | | OUT | /Poni in uscita il registro A sulla |
| 003 011 | 001 | | 001 | /porta di uscita 001 |
| 003 012 | 303 | | JMP | /Salta indietro a REPEAT a |
| 003 013 | 004 | | REPEAT | /LO = 004 e |
| 003 014 | 003 | | Ø | /HI = 003 e ripeti |

**Passo 1**

Nel circuito mostrato per questo esperimento, il buffer/latch 8212 è il registro di istruzione d'interruzione, il generatore di impulsi provoca l'interruzione e l'indicatore a LED indica lo stato del flip-flop di abilitazione all'interruzione dell'8080A, posto all'interno del chip.

Accendete e spegnete il microcomputer parecchie volte. Notate la condizione dell'indicatore a LED dell'interrupt enable (INTE) ogni volta che il microcomputer è alimentato. L'interruzione è abilitata quando il computer è acceso?

Generalmente non lo è, ma questa non è una regola per tutti i microcomputer basati sull'8080A. Con il microcomputer MMD-1, non vi è nessuna istruzione EI in KEX, perciò non è possibile abilitare l'interruzione anche se KEX fosse eseguita accidentalmente dell'accessione.

**Passo 2**

Eseguite il programma dato nell'esperimento. Qual'è lo stato del flip-flop di abilitazione all'interruzione dell'8080A, quando lo eseguite?

Il flip-flop dovrebbe essere a livello logico 1, cioè è abilitato dal programma.

**Passo 3**

Dopo che il programma è partito (notate che continua ad eseguire un loop), osservate il valore del byte presente alla porta di uscita 001. Scrivetelo nello spazio seguente.

Ora settate un valore 004 sugli switch logici, HGFEDCBA = 00000100 = $004_8$, e premete il generatore d'impulsi d'interruzione. Quale byte appare ora alla porta di uscita 001? Qual'è la condizione dell'indicatore a LED INTE?

*Durante un'interruzione, il registro di istruzione d'interruzione forza un'istruzione ad un byte nel registro istruzione dell'8080A.* In questo caso, l'istruzione era 004. Che cosa fa questa istruzione? Potete fare riferimento al listing del set di istruzioni dell'8080A.

Dopo aver settato gli switch logici sul byte istruzioni, 004, e aver rilasciato il generatore d'impulsi d'interruzione, il valore del byte alla porta 001 è incrementato di uno. L'istruzione 004 incrementa i contenuti del registro B, INR B. L'indicatore a LED dell'interrupt enable è a livello logico 0, dopo che l'interruzione è servita.

**Passo 4**

Resettate il microcomputer ed eseguite di nuovo il programma. Notate i valori dei byte alla porta 001 prima e dopo aver azionato il generatore d'impulsi d'interruzione. L'incremento continua?

Sì. Ricordatevi che quando l'MMD-1 viene resettato per la prima volta, KEX mette in uscita un byte d'indirizzo HI sulla porta di uscita 001. È questo byte che è incrementato ogni volta che eseguite il programma e premete il generatore d'impulsi d'interruzione.

**Passo 5**

All'indirizzo di memoria 003 007, sostituite il byte istruzione MOV A,B con un byte istruzione NOP (000). Settate l'istruzione, 074 sugli switch logici alla porta dell'istruzione d'interruzione ed eseguite il programma. Che cosa succede quando provocate un'interruzione attivando il generatore d'impulsi d'interruzione?

Il byte alla porta di uscita 001 è incrementato di uno. In questo caso, abbiamo eseguito un'istruzione INR A prima di mettere in uscita i contenuti dell'accumulatore sulla porta 001. L'istruzione INR A era stata forzata nel registro istruzioni durante l'interruzione.

**Passo 6**

Sostituite un'istruzione 017 sugli switch logici, resettate il microcomputer, eseguite il programma, e premete e rilasciate andare il generatore d'impulsi d'interruzione. Che cosa succede alla porta di uscita 001?

Il byte di dati viene fatto ruotare di una posizione verso destra.

**Passo 7**

Un'interruzione fa sì che l'8080A accetti un'istruzione ad un byte dalla porta dell'istruzione d'interruzione. La natura dell'istruzione influenza i contenuti dello stack? Se non conoscete la risposta a questa domanda, eseguite il seguente esperimento: settate i contenuti delle locazioni di memoria 003 176 e 003 177 su 000. Ripetete il passo 6 o 7, o entrambi, poi resettate il microcomputer ed esaminate le locazioni di memoria 003 176 e 003 177. Che cosa trovate in queste due posizioni? Potete spiegare perchè?

Abbiamo osservato che i contenuti di entrambe le locazioni rimanevano a 000. L'esecuzione dell'istruzione INR A o di RRC non fa sì che il microcomputer ponga un indirizzo sullo stack. Perciò i contenuti dello stack non cambiano. Le uniche istruzioni che influenzano lo stack sono i richiami di subroutine, comprese le istruzioni di ripristino, i rientri dalla subroutine, le istruzioni PUSH, POP e SPHL.

**Passo 8**

Siete capaci di interrompere il programma solo una volta per ogni esecuzione del programma. Perchè?

L'istruzione JMP a 003 012 provoca un salto indietro alla locazione di memoria 003 004 invece che alla locazione di memoria 003 003.

**Passo 9**

Cambiate l'indirizzo di salto alla locazione 003 013 in LO = 003. Ciò permette al programma di saltare indietro ed abilitare l'istruzione di interrupt una volta ogni loop. Ora settate 017 sugli switch logici e rilasciate il generatore d'impulsi d'interruzione. Che cosa succede? Lasciate così il generatore e osservate che cosa accade.

I dati sono ruotati a caso. Se lasciato andare continuativamente, il generatore d'impulsi applica un livello logico 1 all'ingresso INT dello 8080A, che interrompe continuamente l'esecuzione del programma in corso. L'indicatore di monitor dell'interrupt enable, e tutti i LED alla porta di uscita 001 si accendono.

**Passo 10**

Basandovi sulle vostre osservazioni del passo 10, l'ingresso d'interruzione al chip 8080A è edge triggered o sensibile al livello? Come potete ovviare al problema che avete incontrato nel passo 10?

L'ingresso d'interruzione del chip 8080A, INT, è sensibile al *livello.* Se il pin d'ingresso di questo ingresso è a livello logico 1, avverrà una interruzione, fino a che il flag di interrupt è abilitato. Per ovviare a questo problema, potete inserire un flag positive-edge triggered fra il generatore d'impulsi e l'ingresso INT.

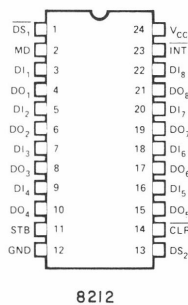
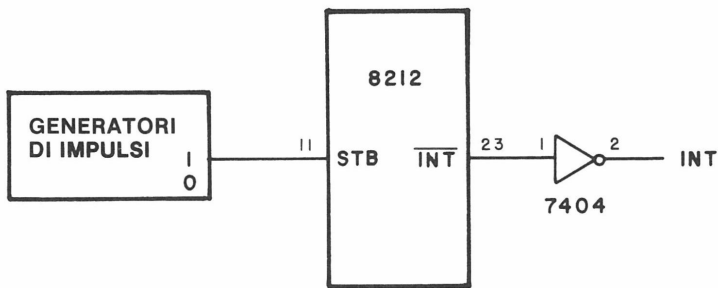
*Conservate l'hardware e il software dati in questo esperimento e passate a quello successivo.*

## ESPERIMENTO N. 8

## COME FORZARE UN'ISTRUZIONE DI RIPRISTINO

### Scopo

Lo scopo di questo esperimento è mostrare l'uso di un flag di interrupt assieme ad un'istruzione di ripristino.

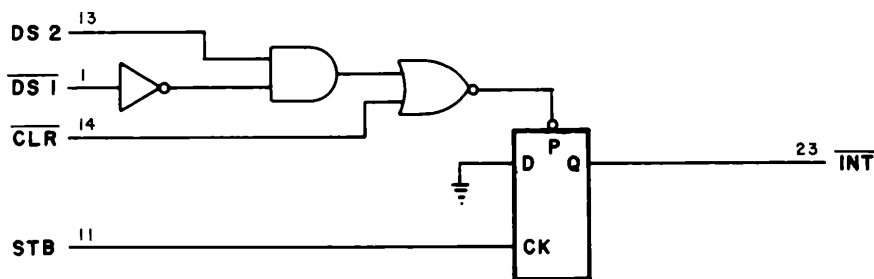
### Configurazione dei pin del circuito integrato

### Schema del circuito

### Programma

Il programma è identico a quello dato nell'Esperimento N. 7.

### Passo 1

Avremo bisogno di apportare le seguenti modifiche al circuito dato nell'Esperimento N.7:

- Scollegare il generatore d'impulsi e l'ingresso INT sul breadboard
- Collegare il generatore d'impulsi all'ingresso STB (pin 11) sul chip 8212
- Collegare l'uscita $\overline{INT}$ (pin 23) sul chip 8212 ad un invertitore, e poi collegare l'invertitore ad INT sul socket del breadboard.

Il buffer/latch 8212 incorpora i seguenti flag di tipo interrupt:

che possono essere usati per fornire un segnale di interrupt al chip 8080A. Ricordatevi che questo circuito è già costruito nel chip 8212; non dovete montarlo sul breadboard.

**Passo 2**

Una volta apportati i necessari cambiamenti hardware, assicuratevi che il vostro programma sia lo stesso di quello dato nell'Esperimento n. 7. Con gli switch logici del registro di istruzione d'interruzione settati sul byte istruzioni 017, eseguite il programma. Supponendo che il byte istruzione all'indirizzo di memoria 003 013 sia 003, dovreste osservare che i dati alla porta di uscita 001 ruotano di una posizione verso destra ogni volta che premete il generatore d'impulsi. È vero?

Sì.

**Passo 3**

Ora eseguirete un semplice programma tramite gli switch logici del registro di istruzione d'interruzione. Eseguite le seguenti operazioni in sequenza:

- Settate gli switch logici su 227. Premete il generatore d'impulsi ed interrompete il microcomputer.
- Poi, settate gli switch logici su 074 ed interrompete il microcomputer una seconda volta.
- Ora settate gli switch logici su 017 ed interrompete il microcomputer una terza volta.

- Infine, continuate ad interrompere il microcomputer parecchie volte.

Quando avete fatto tutto questo, che cosa osservate alla porta di uscita 001?

Si accende un solo indicatore a LED nella posizione D0, e poi ruota verso destra per ogni interruzione.

**Passo 4**

Fin qui, avete dimostrato di poter forzare una varietà di istruzioni ad un byte nel registro istruzione durante un'interruzione. Una volta nel registro istruzione, esse saranno eseguite come regolari byte di istruzioni. Ora dimostreremo come forzare una istruzione di ripristino, RST X.

Inserite il seguente programma nella memoria di lettura/scrittura alle locazioni indicate:

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 020 | 074 | INR A | /Incrementa A |
| 003 021 | 311 | RET | /Rientro dalla subroutine |
| . | | | |
| . | | | |
| 003 050 | 027 | RAL | /Ruota A verso sinistra attraverso il flag di carry |
| 003 051 | 311 | RET | /Rientro dalla subroutine |

Eseguite il programma un'altra volta premendo il tasto RESET e poi G.

**Passo 5**

Settate gli switch logici su 327 al registro di istruzione d'interruzione. Osservate che cosa succede ogni volta che generate un'interruzione al microcomputer.

Il byte alla porta di uscita 001 si incrementa di uno, per ogni impulso d'interruzione.

**Passo 6**

Ora cambiate lo switch logico in 357. Agite sul generatore d'impulsi d'interruzione parecchie volte. Che cosa succede?

I dati ruotano verso sinistra di una posizione per ogni avvenuta interruzione.

**Passo 7**

Caricate 000 nelle locazioni di memoria 003 176 e 003 177, che sono i primi due byte nello stack. Con gli switch logici settati a 357 attuate il RESET del microcomputer e poi eseguite il programma. Generate una sola interruzione, poi il RESET del microcomputer ed esaminate queste due locazioni di memoria, che erano prima a 000. Lo stack è stato usato? Scrivete i contenuti di queste due locazioni di memoria nella tabella seguente. Ripetete il passo 7 parecchie volte, annotando ancora i risultati nello spazio seguente.

| | Contenuti di memoria | |
|---|---|---|
| Tentativo | 003 177 | 003 176 |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |

Abbiamo osservato i seguenti byte dello stack:

| | Contenuti di memoria | |
|---|---|---|
| Tentativo | 003 177 | 003 176 |
| 1 | 003 | 006 |
| 2 | 003 | 012 |
| 3 | 003 | 007 |
| 4 | 003 | 003 |
| 5 | 003 | 003 |
| 6 | 003 | 010 |

Che cosa potete concludere da questa informazione?

I byte dello stack indicavano l'indirizzo di memoria dell'istruzione seguente da eseguire dopo un rientro dalla routine di servizio interruzione. In tutti i casi, questo indirizzo di memoria era dentro i limiti del nostro programma originale. Questo era da prevedere, dato che un rientro dalle subroutine d'interruzione dovrebbe sempre puntare indietro al loop del programma. I contenuti dello stack possono essere diversi dai nostri, ma tutti gli indirizzi dovrebbero essere contenuti all'interno del loop.

*Conservate l'hardware e il software per l'esperimento seguente.*

## ESPERIMENTO N. 9

## TEMPO DI RISPOSTA ALL'INTERRUZZIONE

### Scopo

Lo scopo di questo esperimento è esaminare la risposta di un sistema 8080 alle interruzioni, quando il microcomputer ha altri compiti da eseguire.

### Schema del circuito

Il circuito è identico a quello dato nell'Esperimento n. 8. Vedere la pagina seguente.

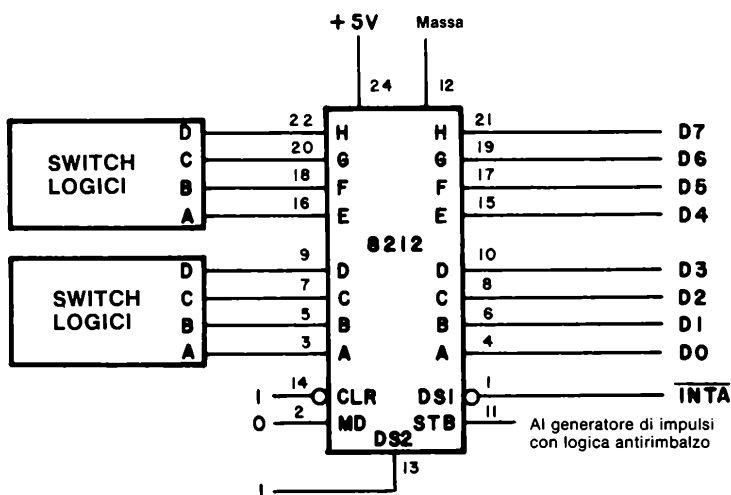
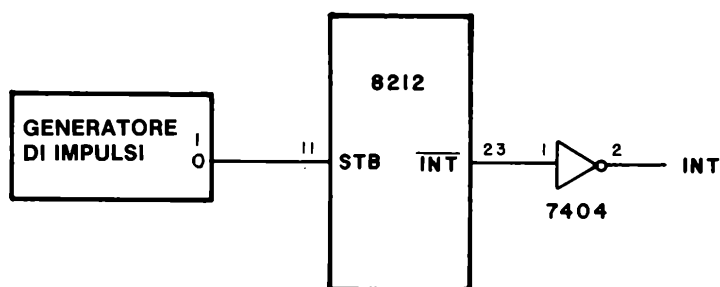
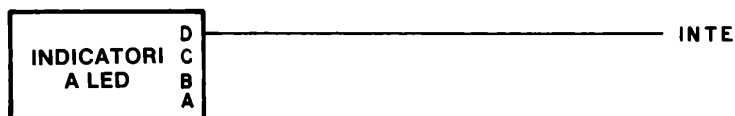
### Programma: Main task

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | 061 | LXI SP | /Carica lo stack pointer con il |
| 003 001 | 200 | 200 | /byte d'indirizzo LO e il |
| 003 002 | 003 | 003 | /byte d'indirizzo HI |
| 003 003 | 373 | LOOP, EI | /Abilita l'interruzione |
| 003 004 | 315 | CALL | /Richiama la subroutine DELAY a |
| 003 005 | 100 | DELAY | /LO = 100 e |
| 003 006 | 003 | ∅ | /HI = 003 |
| 003 007 | 000 | NOP | /Nessuna operazione |
| 003 010 | 000 | NOP | /Nessuna operazione |
| 003 011 | 000 | NOP | /Nessuna operazione |
| 003 012 | 303 | JMP | /Salta indietro a LOOP a |
| 003 013 | 003 | LOOP | /LO = 003 |
| 003 014 | 003 | ∅ | /e HI = 003 |

### Subroutine di ritardo

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 100 | 016 | DELAY, MVI C | /Carica C con |
| 003 101 | 001 | 001 | /il byte di dati 001 |
| 003 102 | 315 | TIME, CALL | /Richiama la sub. KEX TIMEOUT a |
| 003 103 | 277 | TIMEOUT | /LO = 277 e |
| 003 104 | 000 | ∅ | /HI = 000 |
| 003 105 | 015 | DCR C | /Decrementa C di uno |
| 003 106 | 302 | JNZ | /Il registro C è = 000? Se no, salta a |
| 003 107 | 102 | TIME | /TIME a LO = 102 e |
| 003 110 | 003 | ∅ | /HI = 003. Altrimenti, passa /all'istruzione seguente |
| 003 111 | 311 | RET | /Fatto. Rientro dalla subroutine |

**Schema del circuito**

Lo schema del circuito è fondamentalmente identico a quello dell'Esperimento n. 7.

nel quale l'ingresso STB (pin 11) è ora collegato ad un generatore d'impulsi con logica antirimbalzo e l'uscita $\overline{INT}$ (23) è collegata ad un invertitore 7404 che è legato all'ingresso INT del microcomputer MMD-1.

Come prima, un indicatore a LED è usato per visualizzare l'uscita INTE del chip 8080A,

**Compito d'interruzione**

| | | | |
|---|---|---|---|
| 003 020 | 365 | PUSH PSW | /Salva l'accumulatore e i flag |
| 003 021 | 004 | INR B | /Incrementa il registro B |
| 003 022 | 170 | MOV A,B | /Sposta B verso A |
| 003 023 | 323 | OUT | /Poni in uscita i contenuti dell'acc. |
| 003 024 | 001 | 001 | /sul dispositivo 001 |
| 003 025 | 361 | POP BSW | /Memorizza di nuovo accumulatore e /flag |
| 002 026 | 311 | RET | /Fatto. Rientro dalla subroutine |

**Passo 1**

Caricate il Programma nella memoria di lettura/scrittura. Osservate che il software di gestione d'interruzione contiene ora un'istruzione PUSH e una POP. Perchè esse sono necessarie?

Le altre routine usano il registro dell'accumulatore, perciò dobbiamo conservarlo. Quando il microcomputer viene interrotto, non sappiamo qual'è il contenuto dell'accumulatore.

**Passo 2**

Eseguite il software. Ora attivate il generatore d'impulsi d'interruzione. Osservate un ritardo significativo fra l'incremento della porta di uscita 001 e la pressione del tasto?

No. Non abbiamo osservato nessun ritardo, e non ce ne dovrebbe essere. La subroutine DELAY provoca solo un ritardo di 10 millisecondi.

**Passo 3**

Cambiate la costante di tempo alla locazione di memoria 003 101 con i valori dati nella tabella seguente. Come avete fatto nella 3º Fase dell'Esperimento n. 2, applicate 10 impulsi di clock dal generatore d'impulsi il più velocemente possibile (Tentativo 1). Inserite il numero di conteggi che osservate nella tabella seguente. Se volete azzerare B, caricate il software seguente:

| | | | |
|---|---|---|---|
| 003 050 | 006 | MVI B | /Sposta nel registro B il |
| 003 051 | 000 | 000 | /byte di dati 000 |
| 003 052 | 311 | RET | /Rientro dalla subroutine |

Posizionate gli switch logici sull'istruzione di ripristino, 357, attivate il generatore d'impulsi d'interruzione e osservate che la porta di uscita 001 viene azzerata. Riportate gli switch logici a 327 e continuate con l'esperimento:

| Byte ottale di timing | Ritardo di tempo | Tentativo 1 Passo 3 | Tentativo 2 Passo 4 |
|---|---|---|---|
| 012 | 100 ms | | |
| 024 | 200 ms | | |
| 062 | 500 ms | | |
| 144 | 1 s | | |
| 310 | 2 s | | |

Abbiamo osservato che il risultato era confrontabile con quello dell'esperimento N. 2 relativo alla risposta dei flag. Perchè le interruzioni sono così lente in questo caso?

Non importa quando avviene l'interruzione, noi possiamo solo riabilitare il flag dopo che il controllo di programma ritorna all'istruzione EI all'indirizzo di memoria LOOP. Se il loop DELAY è di lunga durata, ci vorrà parecchio tempo per ritornare all'istruzione EI.

**Passo 4**

Apportate le seguenti modifiche al software di gestione d'interruzione:

```
003 026     373    EI       /Abilita l'interruzione
003 027     311    RET      /Rientro dalla subroutine
```

Il flag di interrupt è ora riabilitato immediatamente dopo che l'istruzione RETURN è eseguita all'indirizzo di memoria 003 027. Ricordate che l'istruzione di abilitazione all'interruzione fa effetto sempre dopo l'istruzione che la segue.

Ripetete il passo 3 e notate i vostri risultati nella colonna Tentativo 2 della tabella sopra riportata. Che cosa osservate?

Dovreste osservare che il tempo di risposta del microcomputer è ora essenzialmente indipendente dal lungo ritardo, dato che l'istruzione di abilitazione all'interruzone nel software di gestione d'interruzione riporta indietro l'interruzione a seguire l'esecuzione dell'istruzione seguente. Non dobbiamo aspettare più il lungo ritardo per ritornare all'istruzione EI all'indirizzo di memoria LOOP.

**Passo 5**

L'istruzione EI all'indirizzo di memoria 003 003 è necessaria? Eliminatela sostituendola con un'istruzione NOP (000) e tentate di manovrare il generatore d'impulsi d'interruzione. Che cosa succede?

Niente. L'interruzione non è mai resa attiva, perciò non siamo in grado di richiamare la nostra subroutine di gestione d'interruzione.

Ricordate: se vogliamo usare l'interruzione, abilitiamola! Questo esperimento dovrebbe mostrarvi che le interruzioni, se usate propriamente, possono essere servite immediatamente. Non vi è ritardo di tempo fra la richiesta del servizio e la risposta del microcomputer. Questo non era il caso dei flag dell'Esperimento n. 1.

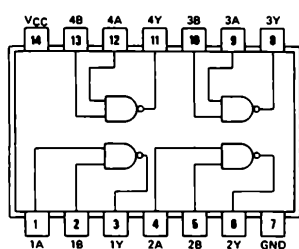
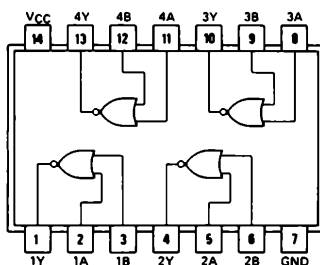
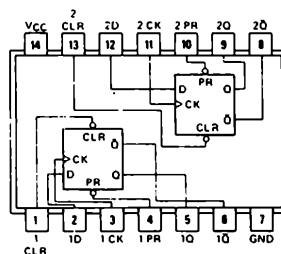
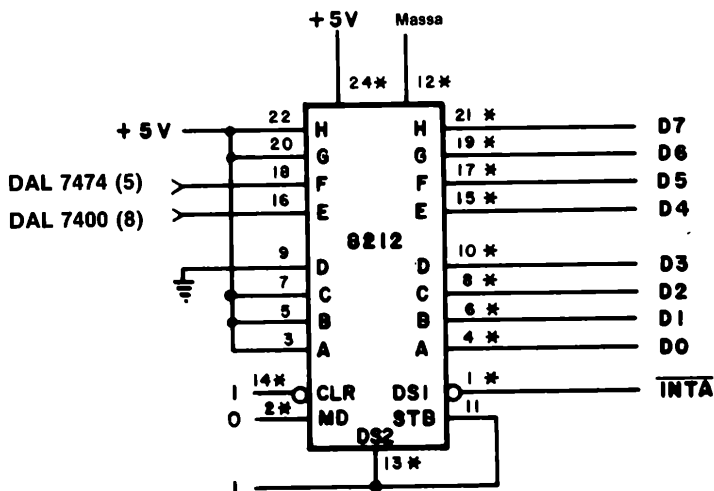
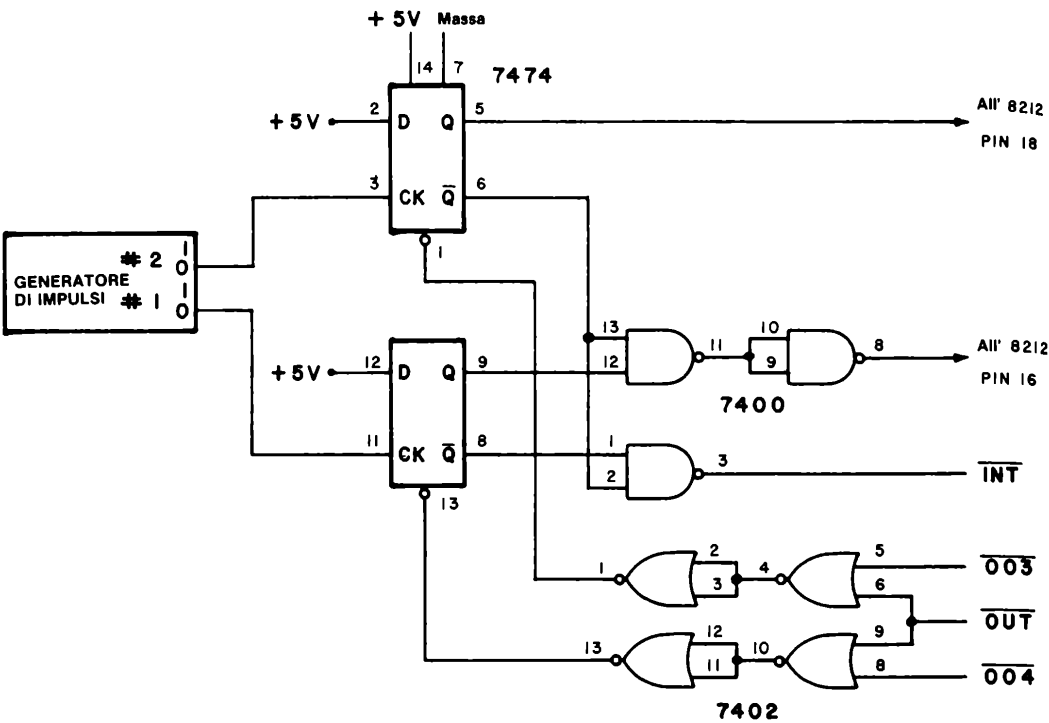
*Conservate il vostro hardware e il vostro software per l'esperimento successivo. Sebbene non venga mostrato nell'Esperimento n. 10, continuate ad usare un indicatore a LED per visualizzare lo stato logico dell'uscita INTE del chip 8080A.*

## ESPERIMENTO N. 10

## SEMPLICI INTERRUZIONI CON PRIORITA'

### Scopo

Lo scopo di questo esperimento è esaminare l'implementazione di un semplice schema di interruzione con livelli diversi di priorità.

### Configurazioni dei pin dei circuiti integrati

### Schemi dei circuiti

**Programma: Main task**

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 000 | 061 | LXI SP | /Carica lo stack pointer con |
| 003 001 | 350 | 350 | /LO = 350 e |
| 003 002 | 003 | 003 | /HI = 003 |
| 003 003 | 373 | LOOP, EI | /Abilita l'interruzione |
| 003 004 | 303 | JMP | /Salta indietro a LOOP a |
| 003 005 | 003 | LOOP | /LO = 003 e |
| 003 006 | 003 | ∅ | /HI = 003 |

**Subroutine di servizio del dispositivo a priorità inferiore**

| | | | |
|---|---|---|---|
| 003 020 | 323 | OUT | /Azzera il flag di interrupt esterno |
| 003 021 | 004 | 004 | /Codice dispositivo 004 |
| 003 022 | 074 | INR A | /Incrementa l'accumulatore |

| | | | |
|---|---|---|---|
| 003 023 | 323 | OUT | /Poni in uscita i contenuti dell'acc. |
| 003 024 | 001 | 001 | /sulla porta di uscita 001 |
| 003 025 | 311 | RET | /Rientro dalla subroutine |

**Subroutine di servizio del dispositivo a priorità superiore**

| | | | |
|---|---|---|---|
| 003 040 | 323 | OUT | /Azzera il flag di interrupt esterno |
| 003 041 | 003 | 003 | /Codice dispositivo 003 |
| 003 042 | 017 | RRC | /Rotazione dei contenuti<br>/dell'accumulatore verso sinistra |
| 003 043 | 323 | OUT | /Poni in uscita i contenuti dell'acc. |
| 003 044 | 001 | 001 | /sulla porta di uscita 001 |
| 003 045 | 311 | RET | /Rientro dalla subroutine |

**Passo 1**

Ricablate l'hardware. Osservate che alcuni dei collegamenti, quelli segnati con un asterisco, sono gli stessi di quelli usati negli esperimenti precedenti.

Caricate il software nella memoria di lettura/scrittura e provatelo attentamente.

**Passo 2**

Eseguite MAIN TSK iniziando dalla locazione di memoria 003 000. L'indicatore a LED può essere usato per controllare lo stato dell'abilitazione all'interruzione (INTE). Esso dovrebbe essere a livello logico 1 non appena il software ha inizio.

Premete e rilasciate il generatore d'impulsi #1 parecchie volte. Osservate cambiamenti alla porta di uscita 001?

Abbiamo osservato che il valore alla porta di uscita 001 è incrementato di uno per ogni volta che agisce il generatore d'impulsi # 1.

**Passo 3**

Premete e rilasciate il generatore d'impulsi #2 parecchie volte. Che cosa succede ora alla porta di uscita 1?

Abbiamo osservato che i dati hanno ruotato verso destra di una posizione ogni volta che agiva il generatore d'impulsi #2. Se non osservate questi risultati nel passo 2 e 3, tornate indietro e controllate sia l'hardware che il software. Il software inserito per i dispositivi a priorità superiore e inferiore è stato usato solo per provare la vostra interfaccia. Nell'esperimento verrà ora usato uno nuovo programma.

**Passo 4**

Ora cambierete il software del dispositivo a priorità superiore, cioè la subroutine di servizio a 003 040, in modo che occupa più tempo per eseguire il suo compito. Caricate il software seguente nella memoria di lettura/scrittura:

**Subroutine di servizio del dispositivo a priorità superiore**
(Prende il posto della subroutine di servizio precedente)

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 040 | 323 | OUT | /Azzera il flag di interrupt esterno |
| 003 041 | 003 | 003 | /Codice dispositivo 003 |
| 003 042 | 365 | PUSH PSW | /Salva l'acc. ed i flag nello stack |
| 003 043 | 305 | PUSH B | /Salva i registri B e C nello stack |
| 003 044 | 000 | NOP | /Nessuna operazione |
| 003 045 | 006 | MVI B | /Carica il registro B con |
| 003 046 | 010 | 010 | /il byte di dati 010 |
| 003 047 | 174 | AGAIN, MOV A,H | /Sposta H verso A |
| 003 050 | 074 | INR A | /Incrementa l'accumulatore |
| 003 051 | 147 | MOV H,A | /Sposta A verso H |
| 003 052 | 323 | OUT | /Poni in uscita i contenuti dell'acc. |
| 003 053 | 000 | 000 | /sulla porta di uscita 000 |
| 003 054 | 315 | CALL | /Richiama la subroutine DELAY a |
| 003 055 | 100 | DELAY | /LO = 100 e |
| 003 056 | 003 | Ø | /HI = 003 |
| 003 057 | 005 | DCR B | /Decrementa il registro B |
| 003 060 | 302 | JNZ | /Il registro B è = 000? Se no, salta a |
| 003 061 | 047 | AGAIN | /AGAIN a LO = 047 e |
| 003 062 | 003 | Ø | /HI = 003. Altrimenti, passa /all'istruzione successiva |
| 003 063 | 301 | POP B | /Sì, B = 000. Ripristina i registri B e C |
| 003 064 | 361 | POP PSW | /Ripristina l'accumulatore e i flag |
| 003 065 | 000 | NOP | /Nessuna operazione |
| 003 066 | 311 | RET | /Rientro dalla subroutine |

**Subroutine di ritardo**

| | | | |
|---|---|---|---|
| 003 100 | 016 | DELAY, MVI C | /Carica C con |
| 003 101 | 144 | 144 | /il byte di dati 144 |

| | | | |
|---|---|---|---|
| 003 102 | 315 | TIME, CALL | /Richiama la sub. KEX TIMEOUT a |
| 003 103 | 277 | TIMEMOUT | /LO = 277 e |
| 003 104 | 000 | Ø | /HI = 000 |
| 003 105 | 015 | DCR C | /Decrementa C di uno |
| 003 106 | 302 | JNZ | /Il registro C è = 000? Se no, salta a |
| 003 107 | 102 | TIME | /TIME a LO = 102 e |
| 003 110 | 003 | Ø | /HI = 003. Altrimenti, passa /all'istruzione successiva |
| 003 111 | 311 | RET | /Fatto. Rientro dalla subroutine |

La subroutine DELAY genera un ritardo di tempo di circa un secondo. Essa richiama la subroutine TIMEOUT di 10 millisecondi nella KEX EPROM.

**Passo 5**

Eseguite il software del MAIN TASK. Generate un'interruzione del dispositivo di priorità inferiore usando il generatore d'impulsi ≠ 1. Questo fa sì che l'istruzione RST 2 venga mandata al microprocessore 8080. Che effetto ha questo fatto?

Dovrebbe incrementare ancora il conteggio presente alla porta di uscita 001. Noi abbiamo cambiato il software per questa routine di servizio d'interruzione.

**Passo 6**

Fate agire il generatore d'impulsi ≠ 2 per generare un'interruzione del dispositivo a priorità superiore. Osservate la porta di uscita 000. Che cosa succede?

Il conteggio incrementa di otto volte in un periodo di tempo di circa otto secondi.

**Passo 7**

Tentate di usare il dispositivo a priorità inferiore (generatore d'impulsi ≠ 1) nel periodo in cui viene eseguita la subroutine del dispositivo a priorità superiore, cioè durante il tempo in cui gli indicatori a LED sono sottoposti ad incremento. Potete attuare questa operazione attivando dapprima il generatore d'impulsi ≠ 2 e poi, velocemente, attivando il generatore d'impulsi ≠ 1 parecchie volte. Che cosa succede?

Il generatore d'impulsi d'interruzione del dispositivo a priorità inferiore non ha effetto, se siete abbastanza veloci da essere capaci di interrompere *durante* l'esecuzione del software di priorità superiore.

Se volete rallentare ancora di più la subroutine di servizio del dispositivo a priorità superiore, cambiate il byte di timing alla locazione di memoria 003 107 in 310, che corrisponde ad un ritardo di tempo di due secondi.

**Passo 8**

Perchè il dispositivo a priorità inferiore non può interrompere quello a priorità superiore? È a causa dell'hardware di priorità.

Il dispositivo a priorità inferiore non può interrompere quello a priorità superiore, perchè il software di gestione della priorità superiore non riabilita l'interruzione fino a che il controllo non è tornato al programma MAIN TASK. Questo è completamente indipendente dall'hardware; esso illustra un problema potenziale: *le interruzioni non possono essere usate a meno che l'interruzione non venga prima abilitata.* La priorità può essere stabilita o in hardware o in software.

*Conservate l'hardware e il software per i due esperimenti successivi.*

## ESPERIMENTO N. 11

## TEMPORIZZAZIONE RELATIVA ALLA INTERRUZIONE CON PRIORITA'

**Scopo**

Lo scopo di questo esperimento è studiare la relazione temporale fra i dispositivi a priorità superiore ed inferiore e come viene assegnata la priorità.

**Schema del circuito**

Il circuito è identico a quello dato nell'Esperimento n. 10.

**Programma: Subroutine di servizio del dispositivo a priorità inferiore**

| Indirizzo di memoria | Byte di istruzione | Codice mnemonico | Commento |
|---|---|---|---|
| 003 020 | 303 | JMP | /Se vi è l'interruzione salta a |
| 003 021 | 150 | 150 | /LO = 150 e |
| 003 022 | 003 | 003 | /HI = 003 |
| 003 150 | 323 | OUT | /Azzera il flag di interrupt |
| 003 151 | 004 | 004 | |
| 003 152 | 365 | PUSH PSW | /Salva l'accumulatore ed i flag |
| 003 153 | 305 | PUSH B | /Salva la coppia di registri B |
| 003 154 | 000 | NOP | /Nessuna operazione |
| 003 155 | 006 | MVI B | /Carica B con |
| 003 156 | 010 | 010 | /Dati = 010 = 8 decimale |
| 003 157 | 056 | MVI L | /Carica L con |
| 003 160 | 200 | 200 | /Dati = $10000000_2$ |
| 003 161 | 175 | LOOPIT, MOV A,L | /Sposta L verso A |
| 003 162 | 007 | RLC | /Rotazione verso sinistra |
| 003 163 | 157 | MOV L,A | /Sposta A verso L |
| 003 164 | 323 | OUT | /Poni in uscita sulla porta 001 |
| 003 165 | 001 | 001 | /il contenuto dell'accumulatore |
| 003 166 | 315 | CALL | /Richiama la subroutine DELAY |
| 003 167 | 100 | DELAY | |
| 003 170 | 003 | Ø | |
| 003 171 | 005 | DCR B | /Decrementa B |
| 003 172 | 302 | JNZ | /B è = 0? Se no, torna indietro col loop. |
| 003 173 | 161 | LOOPIT | /Se sì, passa all'istruzione seguente |
| 003 174 | 003 | Ø | |

| | | | |
|---|---|---|---|
| 003 175 | 301 | POP B | /Ripristino della coppia di registri B |
| 003 176 | 361 | POP PSW | /Ripristino dell'accumulatore e dei flag |
| 003 177 | 000 | NOP | /Nessuna operazione |
| 003 200 | 311 | RET | /Rientro |

**Passo 1**

Nell'esperimento precedente, avete osservato che il dispositivo a priorità superiore potrebbe monopolizzare il tempo del microcomputer e non permettere al dispositivo a priorità inferiore di interrompere.

Caricate il nuovo software, per rendere la subroutine di servizio del dispositivo a priorità inferiore abbastanza lenta.

**Passo 2**

Eseguite il software iniziando dal MAIN TASK, locazione di memoria 003 000. Provatelo generando prima un'interruzione dal dispositivo a priorità superiore e poi da quello a priorità inferiore. Che cosa succede?

Il dispositivo a priorità superiore continua a incrementare lentamente un conteggio; l'altro ruota di un bit verso sinistra, lentamente.

**Passo 3**

Provocate un'interruzione di priorità superiore con il generatore d'impulsi; mentre il software di priorità superiore opera, mandate impulsi per il software di priorità inferiore. Che cosa succede?

Il dispositivo a priorità superiore continua ad operare; il software a priorità inferiore inizia quando il software a priorità superiore è finito.

**Passo 4**

Provocate un'interruzione di priorità inferiore e poi provocatene una di priorità superiore usando i generatori d'impulsi appropriati. Che cosa osservate?

Il software a priorità inferiore continua ad operare. Quando è finito, opera il software a priorità superiore.

**Passo 5**

Apportate le seguenti modifiche nel software del MAIN TASK:

```
.
.
003 003    373          EI       /Abilita l'interruzione
003 004    000    LOOP, NOP      /Nessuna operazione
003 005    303         JMP       /Salta a LOOP
003 006    004        LOOP
003 007    003          Ø
```

**Passo 6**

Potete eseguire più di un'interruzione con questo software? Perchè si o perchè no?

Non potremmo. L'istruzione di abilitazione all'interruzione è stata rimossa dal loop. Una volta usata non possiamo ritornarvi, a meno chè non resettiamo il microcomputer.

**Passo 7**

Aggiungete le due istruzioni di abilitazione all'interruzione (EI) alle routine di servizio interruzione superiore e inferiore come segue:

```
.
.
003 065    373    EI    /Abilita l'interruzione
.
.
.
.
003 177    373    EI    /Abilita l'interruzione
.
.
```

**Notate che i byte istruzioni precedenti a queste due locazioni erano NOP.**

**Passo 8**

Ripetete i passi 3 e 4 di questo esperimento. I risultati sono gli stessi? Perchè?

Sì. Ora riabilitiamo l'interrupt alla fine di ogni subroutine di servizio.

**Passo 9**

Spostate l'istruzione di abilitazione all'interruzione nel software di servizio a priorità inferiore da 003 177 a 003 154. Per fare questo, fate i seguenti cambiamenti:

```
   .
   .
003 154      373      EI      /Abilita l'interruzione
   .
   .
   .
003 177      000      NOP     /Nessuna operazione
   .
   .
```

**Passo 10**

Ripetete il passo 3. Il risultato è lo stesso?

Sì. Non è stato osservato nessun cambiamento.

**Passo 11**

Generate un'interruzione a priorità inferiore usando un generatore d'impulsi. Quando il bit a livello logico 1 ha ruotato verso il centro, generate un'interruzione a priorità superiore. Che cosa succede? Perché?

L'interruzione a priorità superiore interrompe il software d'interruzione a priorità inferiore, esegue l'operazione di incremento e poi rimanda il controllo al software a priorità inferiore. L'interruzione è ora abilitata all'inizio della routine d'interruzione di priorità inferiore, permettendo così ad altri dispositivi di priorità maggiore di interromperla.

**Passo 12**

Eseguite di nuovo il passo 11 ma generate molte interruzioni a priorità superiore durante il tempo di servizio a priorità inferiore. Che cosa succede?

Il dispositivo a priorità superiore è sempre servito.

## ESPERIMENTO N. 12

## INTERRUZIONI SIMULTANEE

### Scopo

Lo scopo di questo esperimento è esaminare l'operazione di interruzione simultanea.

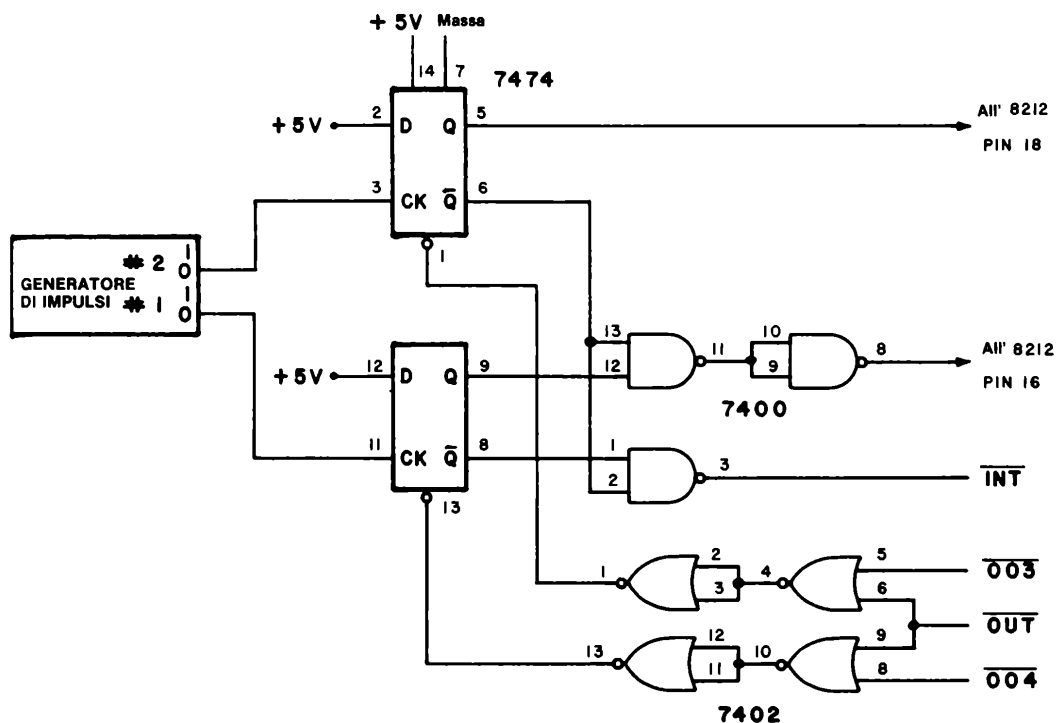
### Schema del circuito

Il circuito è identico a quello usato nell'Esperimento n. 10 e 11.

### Programma

Il software è lo stesso usato nell'Esperimento n. 11.

### Passo 1

Sono qui riprodotti i circuiti usati nell'Esperimento n. 10, atti a generare interruzioni:

Notate che un flip-flop ha la sua uscita Q applicata direttamente al chip 8212 (pin 18), mentre l'uscita Q dell'altro flip-flop passa attraverso dei gate NAND e va all'8212 (pin 16). In questo modo, le istruzioni di ripristino RST 2, 327, o RST 4, 347, vengono generate. Usando lo schema, riempite la tabella della verità seguente:

| DISPOSITIVO A PRIORITA' SUPERIORE | | DISPOSITIVO A PRIORITA' INFERIORE | | chip 8212 | | |
|---|---|---|---|---|---|---|
| Q | $\overline{Q}$ | Q | $\overline{Q}$ | pin 18 | pin 16 | Condizione |
| 0 | 1 | 0 | 1 | | | |
| 0 | 1 | 1 | 0 | | | |
| 1 | 0 | 0 | 1 | | | |
| 1 | 0 | 1 | 0 | | | |

I nostri risultati, dati in ordine verticale, sono i seguenti:

| chip 8212 | | |
|---|---|---|
| pin 18 | pin 16 | Condizione |
| 0 | 0 | Nessuna interruzione |
| 0 | 1 | Interruzioni del dispositivo a priorità inferiore |
| 1 | 0 | Interruzioni del dispositivo a priorità superiore |
| 1 | 1 | Interruzione simultanea |

**Passo 2**

La tabella della verità suddetta indica che cosa succederebbe se sia il dispositivo di priorità superiore che quello inferiore tentassero di interrompere nello stesso momento? Che cosa succederebbe?

Sì. Mostra che il dispositivo a priorità superiore scavalcherebbe l'altro e farebbe sì che venisse eseguita la subroutine di servizio di priorità superiore.

**Passo 3**

Resettate il microcomputer, non iniziate il programma. Spostate il filo dall'uscita a livello logico 0 del generatore d'impulsi ≠1 e mettetelo nell'uscita a livello logico 0 del generatore d'impulsi ≠ 2. Entrambe le interruzioni saranno generate ora dallo stesso generatore d'impulsi.

Ora iniziate il programma. Lasciate che finisca prima di procedere con l'esperimento

**Passo 4**

Osservate attentamente gli indicatori a LED alle porte 000 e 001.

Premete e rilasciate il generatore d'impulsi d'interruzione, # 2. Che cosa succede?

Abbiamo osservato che il software a priorità superiore (accensione dei LED) è iniziato; quando è finito, inizia il software a priorità inferiore (rotazione di un bit) e completa il suo compito.

**Passo 5**

Ripetete il Passo 4 ancora cinque volte. Il software del dispositivo a priorità superiore inizia sempre la sequenza?

Sì.

**Passo 6**

Resettate il microcomputer. Non iniziate il programma. Rimettete il filo, che avete spostato nel passo 3, sull'uscita di livello logico 0 del generatore d'impulsi # 1. Provocate un'interruzione a priorità inferiore e, a metà dell'operazione della routine di servizio, generate di nuovo un'interruzione a priorità inferiore. Che cosa succede?

La seconda routine di servizio è completata. La prima routine ha proseguito fino a metà e poi ha iniziato a ruotare di nuovo all'inizio.

**Passo 7**

Potete spiegare che cosa avete osservato nel passo 6?
Nota: Il registro L non viene conservato sullo stack, in questo programma.

La seconda interruzione a priorità inferiore ha interrotto il software di servizio della prima interruzione a priorità inferiore. Dato che il registro L non è memorizzato nello stack, viene lasciato al momento della seconda interruzione in uno stato sconosciuto. Quando la prima routine d'interruzione tenta di usare il registro L, esso non è nello stesso stato in cui era prima che avvenisse la seconda interruzione.

Che soluzioni suggerireste per il problema presentatosi nel passo 7?

Suggeriamo:

1. Azzerare il flag di richiesta di servizio interruzione del dispositivo a priorità inferiore alla fine della routine di servizio.

2. Salvare nello stack tutti i registri che vanno usati nella routine.

Non dovreste mai permettere ad un dispositivo d'interruzione di interrompere la propria routine di servizio software. Se lo fa, il dispositivo esterno opera troppo velocemente per il microcomputer; il software deve essere semplificato per renderlo veloce. Perchè? Perchè mentre entriamo nel software di servizio della seconda interruzione, una terza interruzione interromperà la seconda, ecc. Non saremo mai in grado di completare nessuna routine di servizio.

## DOMANDE RIEPILOGATIVE

Le seguenti domande vi aiuteranno a rivedere l'uso dei flag e delle interruzioni.

1. Che tipi di dispositivi si possono usare come flag e perchè i flag sono importanti?

2. Quali istruzioni del set dell'8080A possono essere usate per rivelare le condizioni interne dei flag?

3. Quali vantaggi ha un'interruzione rispetto al flag interrogato ciclicamente?

4. Quali sono i tre tipi di interruzione e quale tipo è usato nei sistemi a microcomputer dell'8080A?

5. Che tipi di istruzioni sono più utili con le interruzioni dell'8080A? Come lavorano?

6. Come si presenta una tipica subroutine di servizio interruzione?

7. Che cos'è un'interruzione con priorità?

8. Quali sono alcuni dei potenziali problemi riguardanti l'uso delle interruzioni?

**RISPOSTE**

1. I flag possono essere switch, flip-flop, contatori, registri a scorrimento, o memorie. Generalmente, qualunque dispositivo bistabile si può qualificare come flag. I flag di solito si usano per indicare che sono cambiate delle condizioni e per sincronizzare le operazioni di flusso e di controllo dei computer.

2. Tutte le istruzioni condizionate, cioè salti, richiami, e rientri, sono utili, sebbene i salti siano quelli più usati. Altre istruzioni utili sono le rotazioni, AND, OR, ecc., ma non rivelano le condizioni dei flag.

3. La velocità. Le interruzioni sono lette generalmente entro microsecondi, mentre i flag interrogati ciclicamente possono impiegare molto, molto di più, a seconda del software.

4. Le interruzioni possono essere su di una sola linea, a più livelli e vettorizzate. Vedere la Figura 23-7 e il testo relativo.

5. Le istruzioni di ripristino, RST X, sono generalmente le più utili. Esse sono richiami ad un byte che, quando vengono eseguite, provocano un indirizzo di rientro che viene posto nello stack. Il computer poi crea un vettore all'indirizzo della subroutine specificato dall'istruzione di ripristino. RST X richiama una subroutine a 000 0X0. Per finere la subroutine deve essere usata un'istruzione di rientro.

6. Comprende istruzioni PUSH, POP, EI e RET oltre alle routine software di servizio interruzione. Vedere Figura 23-10.

7. Un'interruzione con priorità è una interruzione nella quale esiste una priorità prestabilita per l'ordine nel quale le interruzioni vengono servite dal computer. La priorità può essere fissata nell'hardware o nel software.

8. I grossi problemi sono la determinazione del tempo e la priorità. Questi argomenti sono stati trattati dettagliatamente quasi alla fine di questo Capitolo.

# APPENDICE 1

# RIFERIMENTI

1. *The Compact Edition of the Oxford English Dictionary,* Oxford Univ. Press, 1971.

2. Rudolf F. Graf, *Modern Dictionary of Electronics,* Howard W. Sams & Company, Inc., Indianapolis, 1972.

3. James Martin, *Telecommunications and the Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

4. Abraham Marcus and John D. Lenk, *Computers for Technicians,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

5. Microdata Corporation, *Microprogramming Handbook,* Santa Ana, California, 1971.

6. J. Blukis and M. Baker, *Practical Digital Electronics,* Hewlett-Packard Company, Santa Clara, California, 1974.

7. Donald E. Lancaster, *TTL Cookbook,* Howard W. Sams & Co., Inc., Indianapolis, 1974

8. H. V. Malmstadt, C. G. Enke, and S. R. Crouch, *Instrumentation for Scientists Series, Module 3. Digital and Analog Data Conversion,* W. A. Benjamin, Inc., Menlo Park, California, 1973-4.

9. H. V. Malmstadt and C. G. Enke, *Digital Electronics for Scientists,* W. A. Benjamin, Inc., New York, 1969.

10. J.D. Lenk, *Handbook of Logic Circuits,* Reston Publishing Company, Inc., Reston, Virginia, 1972.

11. A. James Diefenderfer, *Principles of Electronic Instrumentation,* W. B. Saunders Company, Philadelphia, 1972.

12. P. R. Rony and D. G. Larsen, *Bugbook II. Logic & Memory Experiments Using TTL Integrated Circuits,* E & L Instruments, Inc., Derby, Connecticut, 1974.

13. Robert L. Morris and John R. Miller, Editors, *Designing with TTL Integrated Circuits,* Mc Graw-Hill Book Company, New York, 1971.

14. Charles J. Sippl, *Microcomputer Dictionary and Guide,* Matrix Publishers, Inc., Champagne, Illinois 61820, 1976.

15. Donald Eadie, *Introduction to the Basic Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

16. Texas Instruments Incorporated, *Microprocessor Handbook,* Dallas, Texas, 1975.

17. Charles L. Garfinkel of Keithley Instruments, Inc., is the originator of this definition.

# APPENDICE 2

# DIZIONARIO DEI TERMINI TECNICI DEI BUGBOOKS V E VI

Questa appendice costituisce un piccolo dizionario dei termini tecnici più importanti relativi all'elettronica digitale e ai microcomputer riportati nel Bugbook V e VI. Per una migliore consultazione i termini sono indicati sia in italiano che in inglese (fra parentesi).
Le definizioni sono state tratte dai seguenti testi:

- Rudolf F. Graf, *Modern Dictionary of Electronics,* Howard W. Sams & Co., Inc., Indianapolis, 1972.
- Microdata Corporation, *Microprogramming Handbook,* Santa Ana, California, 1972.
- Donald Eadie, *Introduction to the Basic Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
- Abraham Marcus and John D. Lenk, *Computers for Technicians,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
- Peter R. Rony, David G. Larsen, and Jonathan A. Titus, *Bugbook III. Microcomputer Interfacing Experiments Using the Mark 80 Microcomputer, an 8080 System,* E & L Instruments, Inc., Derby, Connecticut, 1975.

*Abilitare (To enable)* — Permettere il passaggio di un segnale digitale in o attraverso un dispositivo digitale o un circuito. Pag. 11-12.

*Addendo (Addend)* — Una quantità che se aggiunta ad un'altra quantità (chiamata augendo) produce un risultato chiamato somma. Pagg. 18-74 e 20-35.

*Accumulatore (Accumulator)* — Il registro e l'insieme dei circuiti elettronici digitali ad esso associati nell'unità logico/aritmetica (ALU) di un computer in cui vengono eseguite operazioni logiche e aritmetiche. Pagg. 3-7 e 18-8.

*Algebra Booleana (Boolean algebra)* — Un sistema di logica matematica relativo alle classi, alle proporzioni, elementi circuitali on-off, etc., associati da operatori quali l'AND, OR, NOT, XOR, etc. Il nome deriva dal matematico inglese George Boole che introdusse questa logica nel 1847. Pag. 8-2.

*Aritmetica in doppia precisione (Double-precision arithmetic)* — 1) Utilizzazione di due parole di un computer per rappresentare un numero, di solito per ottenere una precisione maggiore di quella che è in grado di fornire una sola parola.
2) Aritmetica usata quando è necessaria più precisione di quella che fornirà la memorizzazione di una sola parola. Pag. 18-36.

| | |
|---|---|
| *Augendo (Augend)* | In un'addizione aritmetica, il numero che aumenta aggiungendovi un altro numero, chiamato addendo. Pagg. 18-35 e 20-35. |
| *Base (Base)* | Indicata anche con il nome di radice. Il numero totale dei diversi simboli utilizzati in un dato sistema di numerazione. Ad esempio, dato che il sistema di numerazione decimale utilizza dieci simboli, la radice è 10. Nel sistema di numerazione ottale, la radice è 8. Nel sistema di numerazione binario, la radice è 2 perchè si hanno solo due simboli (0 e 1). Pag. 1-4. |
| *Binario (Binary)* | Sistema di numerazione utilizzante una base, o radice, due. Nel sistema binario vi sono due digit (0 e 1). Pag. 7-2. |
| *Binario decimale codificato (Binary coded decimal)* | L'abbreviazione è BCD. Si tratta di un sistema di rappresentazione dei numeri in cui ogni digit decimale di un dato numero è espresso da numeri binari. Conosciuto anche come codice 8421. Pag. 12-4. |
| *Bit (Bit)* | Abbreviazione per *BI*nary digi*T*. Unità di informazione eguale all'uno binario, oppure indicazione di uno dei due stati (0 e 1) utilizzati per memorizzare o trasferire una informazione. Pag. 1-3. |
| *Breadboard (Breadboard)* | Qualsiasi ausilio utilizzato per collegare in modo temporaneo tra loro elementi circuitali, al fine di verificare la funzionalità di un dato circuito. Di solito si tratta di piastre o basette recanti componenti o circuiti. Pag. 9-2 |
| *Breadboarding (Breadboarding)* | Azione relativa all'utilizzo di un breadboard per attuare dei collegamenti temporanei in circuiti elettrici. Pag. 9-2 |
| *Buffer (Buffer)* | Elemento di un circuito digitale che può essere utilizzato per gestire un alto fan-out oppure per invertire livelli di ingresso/uscita. Pag. 14-20 |
| *Buffer gate (Buffer gate)* | Circuito digitale che incrementa la capacità di fornire tensione o corrente da parte di un circuito binario. Conosciuto anche come driver (pilota). Pag. 7-13 |
| *Bus (Bus)* | Un canale sul quale vengono trasferite le informazioni digitali, da una delle molte sorgenti ad una delle molte destinazioni. Può aver luogo solo un trasferimento di informazioni alla volta. Mentre avviene tale trasferimento, tutte le altre fonti che sono legate al bus devono essere disabilitate. Pag. 16-12. |
| *Bus di controllo (Control bus)* | Un insieme di segnali che regolano le operazioni del microcomputer, dei dispositivi di I/O e della memoria. Funzionano più come segnali di "traffico" o comandi. Possono anche originarsi nei dispositivi di I/O, generalmente per trasferire a o ricevere segnali dalla CPU. Un insieme unidirezionale di segnali che indicano il tipo di attività - lettura in memoria, scrittura in memoria, lettura di I/O, scrittura in I/O o segnalazione d'interruzione nei processi in corso. Pag. 16-12. |

*Bus di dati bidirezionali (Bidirectional data bus)* Un bus di dati in cui le informazioni digitali possono essere trasferite in una direzione o in un'altra. Riferendosi ad un microcomputer basato sull'8080, il canale di dati bidirezionale tramite il quale i dati sono trasferiti fra la CPU, la memoria e i dispositivi di I/O. Pag. 16-12.

*Bus d'indirizzo (Address bus)* Un bus unidirezionale sul quale appare l'informazione digitale per identificare o una particolare locazione di memoria o un particolare dispositivo di I/O. Il bus d'indirizzo dell'8080 è un gruppo di sedici linee. Pag. 16-12.

*Bus monitor (Bus monitor)* Un display binario, ottale o esadecimale che visualizza i dati che appaiono sul bus di dati bidirezionali. Pag. 17-20.

*Byte (Byte)* Un gruppo di otto bit contigui su cui si opera come su un'unità o che occupano una singola locazione di memoria. Pagg. 2-6 e 18-12.

*Byte di dati (Data byte)* Per un microcomputer basato sull'8080, un byte di dati è un numero binario ad 8 bit trasferito sul bus dati bidirezionale. Pagg. 3-3 e 3-5

*Byte di indirizzo LO (LO address byte)* Gli 8 bit meno significativi di un indirizzo di memoria a 16 bit, per il microprocessor 8080. Pagg. 2-8 e 3-5

*Capacità (Capacitance)* È la capacità di un condensatore o di un sistema conduttore/dielettrico di immagazzinare cariche elettricamente separate quando esiste una differenza di potenziale tra i conduttori. La capacità di un condensatore è definita come il rapporto tra la carica elettrica che è stata trasferita da un elettrodo ad un altro e la differenza di potenziale risultante tra i due elettrodi. L'unità di misura della capacità è il Farad. Pag. 9-11. C (Farads) = Q (Coulombs) / V (Volt).

*Ciclo macchina (Machine cycle)* Una suddivisione elementare del ciclo di una istruzione durante il quale avvengono un insieme di azioni collegate all'interno del microprocessore. Tutte le istruzioni sono combinazioni di uno o più cicli macchina. Pag. 17-16.

*Circuito di gate (Gate circuit)* Circuito che permette il passaggio di un segnale solo quando è presente un impulso di gating. Circuito elettronico con uno o più ingressi e un'uscita in cui un impulso esce sulla linea di uscita se, e solo se, si verificano delle specifiche combinazioni di impulsi alle linee di ingresso. Pag. 14-20

*Circuito di gating (Gating circuit)* Circuito che opera come switch selettivo e permette la conduzione solo durante selezionati intervalli di tempo oppure quando la grandezza del segnale è entro specifici limiti. Pag. 14-21

*Clear (Clear)* Vedere reset. Pag. 11-7

*Clock (Clock)* Detto anche orologio. Qualunque dispositivo che genera uno o più impulsi di clock. Pag. 9-12

| | |
|---|---|
| *Clock* *(Clock)* | Un generatore di impulsi che controlla la temporizzazione di dispositivi "clocked logic" e che regola la velocità cui tali dispositivi operano. Svolge anche un'azione di sincronizzazione tra tutte le operazioni in un sistema digitale. Pag. 11-5 |
| *Clock in tempo reale* *(Real-time clock)* | Si riferisce ad un dispositivo che fornisce interruzioni a intervalli di tempo regolari, spesso due volte maggiori della frequenza di rete c.a. Permette che venga mantenuta una precisa indicazione dello scorrere del tempo e della misura del tempo impiegato. Pagg. 22-10 e 23-20. |
| *Codice ASCII* *(ASCII code)* | American Standard Code for Information Interchange. Codice a sette bit, senza bit di parità, oppure ad otto se con parità. Pag. 12-6 |
| *Codice binario* *(Binary code)* | Codice in cui ogni elemento distingue uno dei due stati. Questi stati sono indicati con i simboli 0 ed 1. Pag. 1-4 |
| *Codice dispositivo* *(Device code)* | In un microcomputer basato sull'8080, il codice a 8 bit per uno specifico dispositivo di ingresso o di uscita. Pagg. 3-3, 3-5 e 17-6. |
| *Codice istruzione* *(Instruction code)* | Un numero binario a 8 bit che codifica un'operazione che il microprocessore 8080 può eseguire. Pag. 18-11. |
| *Codice macchina* *(Machine code)* | Rappresentazione binaria di un'istruzione per computer. Pag. 2-4 |
| *Codice mnemonico* *(Mnemonic code)* | Istruzione scritta in una forma tale da facilitare la comprensione (simbolicamente), ma che può essere convertita in codice macchina. Pag. 2-4 |
| *Codice operazione* *(Operation code)* | Per un microprocessor 8080, il codice ad 8 bit relativo ad una specifica azione. Pag. 3-5 |
| *Codice ottale* *(Octal code)* | Relativo al sistema di numerazione binario a base 8, in cui i numeri da 0 a 7 sono usati per rappresentare i digit ottali da 0 a 7. Pag. 1-5 |
| *Codificare* *(To code)* | Utilizzare un codice spesso costituito da numeri binari, per rappresentare dei singoli caratteri o gruppi di caratteri nell'ambito di un messaggio. Effettuare un cambiamento da un codice ad un altro. Se i codici sono molto differenti, il processo è detto conversione di codice. Pagg. 1-5 e 12-6 |
| *Complemento* *(Complement)* | Complemento di un numero binario. Il complemento di 1 è 0 e il complemento di 0 è 1. Il complemento di 011010 è 100101. Pag. 8-7 |
| *Computer* *(Computer)* | Vedere computer digitale. Pag. 2-2 |
| *Computer digitale* *(Digital computer)* | Un dispositivo elettronico in grado di accettare, memorizzare, manipolare da un punto di vista aritmetico delle informazioni, intese sia come dati che come istruzioni. Le informazioni sono |

trattate in termini di digit codificati secondo il sistema di numerazione binario (0 e 1), e sono rappresentati da due livelli di tensione. Pag. 2-2

| | |
|---|---|
| *Computer sincrono (Synchronous computer)* | Un computer digitale in cui tutte le operazioni ordinarie sono controllate da un clock principale. Pag. 16-17. |
| *Comunicazione (Communication)* | Fornire, inviare, scambiare idee, conoscenze ed informazioni, tramite parole, scritti, segni o segnali. Pag. 1-2 |
| *Condensatore (Capacitor)* | Componente elettronico consistente in due superfici conduttrici separate da materiale isolante o diellettrico (carta, mica, film plastico, dielettrico inorganico). Un condensatore immagazzina energia elettrica, blocca il flusso di corrente continua e permette il flusso di corrente alternata. Tutto ciò essenzialmente in funzione del valore della capacità e della frequenza. Pag. 9-11 |
| *Contatore (Counter)* | Dispositivo in grado di cambiare stato in una specifica sequenza in base al ricevimento di appropriati segnali di ingresso. L'uscita del contatore indica il numero degli impulsi che sono stati applicati (vedere anche divisore). Un contatore deriva dall'insieme di flip-flop con alcune porte (gate). L'uscita di tutti i flip-flop è accessibile per indicare l'esatto conteggio in ogni istante. Pag. 13-2 |
| *Contatore a decade (Decade counter)* | Dispositivo logico con 10 stati stabili, in grado di ciclare attraverso questi stati tramite l'applicazione di dieci impulsi di clock. Un contatore a decade usualmente conta in una sequenza binaria dallo stato 0 allo stato 9 e successivamente ritorna allo stato 0. È anche indicato come contatore-divisore per 10. Pag. A-31 |
| *Contatore binario (Binary counter)* | Interconnessione di flip-flop avente un singolo ingresso, realizzata in modo da permettere un conteggio binario. Ogni volta che un impulso appare all'ingresso, il contatore cambia stato: il numero degli impulsi in ingresso è tabulato in modo da fornire una informazione in uscita in forma binaria. I conteggi possibili sono 2n, dove n è il numero dei flip-flop o degli stadi. Pag. 13-2 |
| *Contatore di programma (Programm counter)* | Il registro a 16 bit nell'8080, che contiene l'indirizzo di memoria del byte d'istruzione successivo che deve essere eseguito in un programma. Pagg. 18-8 e 18-49. |
| *Controller (Controller)* | Uno strumento che mantiene un processo o una condizione ad un livello desiderato o ad uno stato come si determina dal confronto del valore attuale con il valore desiderato. Pag. 16-7. |
| *Controllo (Control)* | Quelle parti di un computer che eseguono istruzioni in sequenza propria, che interpretano istruzioni ed applicano segnali propri. Pag. 16-12. |
| *Conversione di codice (Code conversion)* | Cambiamento della configurazione di bit per un carattere in un codice, nella corrispondente configurazione di bit in un altro codice. Pag. 12-8 |

*Convertitore da analogico a digitale*
*(Analog-to-digital converter)*
Un circuito che converte continuamente una tensione od una corrente in un'uscita digitale. Pag. 22-46.

*Convertitore da digitale in analogico*
*(Digital-to-analog converter)*
Un circuito che cambia un ingresso digitale in una tensione o in una corrente che variano continuamente. Pag. 22-39.

*CPU*
*(CPU)*
Abbreviazione di unità centrale di elaborazione. Pag. 16-13.

*CPU microprocessore*
*(Central processing unit-microprocessor)*
Un solo circuito integrato che realizza trasferimenti di dati, controlli, operazioni di ingresso/uscita, operazioni aritmetiche e logiche eseguendo istruzioni ottenute dalla memoria. Pag. 16-13.

*CPU unità centrale di elaborazione*
*(Central processing unit)*
Chiamata anche processore centrale. Quella parte di un computer che contiene la memoria principale, l'unità aritmetica, e gruppi speciali di registri. Esegue operazioni aritmetiche, controlla l'elaborazione delle istruzioni, fornisce i segnali di timing ed esegue altre operazioni ausiliarie. Pag. 16-13.

*Corsa*
*(Race)*
Condizione che si verifica quando il cambiare lo stato di un sistema richiede un cambiamento in due o più stati variabili. Se lo stato finale subisce un'influenza dallo stato variabile che per primo cambia, la condizione è detta corsa critica. Anche la condizione che esiste quando un segnale si propaga attraverso due o più elementi di memoria durante lo stesso periodo di clock. Pag. 11-11

*Decodifica in assoluto*
*(Absoluting decoding)*
La decodifica di un numero binario per produrre un unico impulso, per selezionare un unico indirizzo di memoria, ecc. Pagg. 17-9 e 17-39.

*Decodificare*
*(To decode)*
Utilizzo di un codice per modificare una precedente codifica. Determinare il significato di un set di impulsi o di segnali logici utilizzati per descrivere una istruzione, un comando od una operazione. Pag. 12-6

*Decodificatore di istruzioni*
*(Instruction decoder)*
Un decodificatore interno al microprocessore 8080 che decodifica il codice istruzione in una serie di azioni che il microprocessore esegue. Pag. 18-11.

*Digit (o nibble)*
*(Nibble)*
Un gruppo di quattro bit contigui su cui si opera come su di una unità o che occupano una sola locazione di memoria. Pag. 18-37.

*Diodo*
*(Diode)*
Semiconduttore a due elettrodi utilizzante le capacità raddrizzatrici di una giunzione pn (diodo a giunzione) oppure di un punto metallico in contatto con un diodo a semiconduttore (diodo a punto di contatto - point contatct diode). Pag. 9-12

| | |
|---|---|
| *Disabilitare (To disable)* | Impedire il passaggio di segnali digitali tramite l'applicazione di un opportuno segnale al terminale di disabilitazione di un dispositivo digitale. Pagg. 11-2 e 14-20 |
| *Display (Display)* | Detto anche indicatore. È un dispositivo che permette di visualizzare un segnale elettronico. Pag. A-24 |
| *Dispositivo a tre stati (Three-state device)* | Un dispositivo logico a semiconduttore in cui vi sono tre possibili stati di uscita: (1) uno stato di livello logico 0, (2) uno stato di livello logico 1 e (3) uno stato in cui l'uscita è , in effetti, scollegata dal resto del circuito e non ha influenza su di esso. Pagg. 19-3 e Capitolo 19. |
| *Dispositivo di I/O (I/O device)* | Dispositivo di ingresso/uscita. Un lettore di schede, un'unità a nastro magnetico, una stampante o dispositivo analogo che trasmette o riceve dati da un computer o da un dispositivo di memorizzazione secondario. In senso più generale, qualunque dispositivo digitale, compreso un singolo circuito integrato, che trasmette dati o riceve dati o impulsi da un computer. Pag. 16-13. |
| *Dispositivo digitale (Digital device)* | Qualsiasi dispositivo che opera o manipola informazioni binarie. Pag. 7-2 |
| *Dispositivo Tri-state® (Tri-state device®)* | Vedi dispositivo a tre stati. Pag. 19-3 e Capitolo 19. |
| *Driver (Driver)* | Detto anche pilota. Elemento di un circuito digitale che viene accoppiato allo stadio di uscita di un circuito per aumentare (pilotare) la capacità di controllo in tensione, in corrente o di fan-out. Per esempio un clock driver viene utilizzato per fornire la necessaria corrente alla linea di clock. Vedere Buffer Gate. Pag. 14-21 |
| *Edge triggered flip-flop (Edge-triggered flip-flop)* | Tipo di flip-flop in cui una minima percentuale di variazione del segnale di clock, in volt/secondi, è una condizione necessaria per determinare il cambiamento dell'uscita. Pag. 13-6 |
| *Elemento astabile (Astable element)* | Elemento a due stati che non ha uno stato stabile. Pag. 15-2 |
| *Elemento bistabile (Bistable element)* | Differente terminologia per indicare il flip-flop. Circuito in cui l'uscita ha due stati stabili raggiungibili in base agli ingressi. Lo stato in uscita rimane anche all'esaurirsi del segnale in ingresso. Pagg. 11-2 e 15-2 |
| *Fan-in (Fan-in)* | Il carico richiesto da un ingresso digitale ad un circuito integrato. Per la famiglia TTL, il carico di ingresso richiesto è normalizzato al valore 1. Un fan-in di 1 corrisponde a 1,6 mA. Pag. 10-16 |
| *Fan-out (Fan-out)* | Il numero di carichi paralleli nell'ambito di una certa famiglia logica, come la TTL, che possono essere pilotati dall'uscita di un circuito logico. Un chip TTL standard ha un fan-out di 10 carichi, il che significa che può pilotare 10 carichi TTL standard ciascuno con un fan-in di 1. Un fan-out di 10 nel caso TTL corrisponde a 16 mA. Pagg. 10-16 e 14-21 |

| | |
|---|---|
| *Flag* *(Flag)* | Un tipo di dispositivo digitale o di registro usato per indicare lo stato di un dispositivo. Può essere azzerato o settato in risposta ad un'operazione. Pag. 23-2. |
| *Flip-Flop* *(Flip-flop)* | Un circuito avente due stati stabili e le capacità di spostarsi da uno stato ad un altro tramite l'applicazione di un segnale di controllo; il flip-flop resta in questo stato anche dopo che il segnale è stato rimosso. Pag. 11-12 |
| *Flip-flop di tipo D* *(D-type flip-flop)* | Il simbolo D indica Delay (ritardo). È un flip-flop la cui uscita è in ritardo rispetto all'ingresso di un impulso di clock; ad esempio, se un 1 logico appare all'ingresso, l'uscita diventerà 1 dopo il successivo impulso di clock. Pag. 11-5 |
| *Forma d'onda digitale* *(Digital waveform)* | Rappresentazione grafica di un segnale digitale, indicante le variazioni dello stato logico in funzione del tempo. Questo tipo di rappresentazione e anche conosciuto con il nome di diagramma di tempo (timing diagram). Pag. 11-9 |
| *Fronte negativo* *(Negative edge)* | La transizione dall'1 logico allo 0 logico in un impulso di clock. Pag. 13-6 |
| *Fronte positivo* *(Positive edge)* | La transizione dallo 0 all'1 logico in un impulso di clock. Pag. 13-6 |
| *Funzione ausiliaria* *(Auxiliary function)* | Tutti i diversi dispositivi elettronici necessari per rendere operativo un circuito digitale: circuiti integrati, resistori, condensatori, etc. Pag. 9-4 |
| *Gated buffer* *(Gated buffer)* | Un circuito driver a bassa impedenza che può essere usato per pilotare una linea per differenziazione di impulsi oppure in multivibratori. In generale si dice di un buffer che è "gated". Pag. 14-20 |
| *Gate (dispositivo di gating)* [*Gate (gating device)*] | Circuito avente due o più ingressi e un'uscita. Uno degli ingressi può essere identificato come ingresso dati ed i rimanenti come ingresso di gating. Lo stato logico degli ingressi di gating determina il passaggio o meno dell'ingresso dati che appare all'uscita. Pag. 14-6 |
| *Gate (dispositivo logico)* [*Gate (logic device)*] | Circuito con due o più ingressi e un'uscita. L'uscita dipende dalla combinazione dei segnali logici in ingresso. Vi sono 4 porte (gate) base AND, OR, NAND, NOR. Pagg. 7-2, 14-6 e 14-15 |
| *Gated driver* *(Gated driver)* | In generale, un driver che è un "gated". Pag. 14-21 |
| *Generatore di impulsi* *(Pulser)* | Switch logico che genera un singolo impulso di clock. Pag. 9-12 |
| *Hardware* *(Hardware)* | I dispositivi meccanici, magnetici, elettronici ed elettrici di cui è composto un computer; l'insieme dei materiali che formano un computer. Pag. 16-7. |

| | |
|---|---|
| *Impulso di clock (Clock pulse)* | Un ciclo logico completo dallo 0 logico all'1 logico ed ancora allo 0 logico (impulso di clock positivi), oppure dall'1 logico allo 0 logico ed ancora all'1 logico (impulso di clock negativo). Pag. A-27 |
| *Impulso di gate (Gate pulse)* | Impulso che abilita (enable) un circuito di gate (gate circuit) a far passare un segnale. L'impulso di porta generalmente ha una durata più lunga del segnale per assicurare la coincidenza. Pag. 14-20 |
| *Impulso di gating (Gating pulse)* | Impulso che modifica o controlla l'operazione di un circuito di gate (gate circuit). Pag. 14-21 |
| *Impulso di selezione dispositivo (Device select pulse)* | Un impulso di clock generato dal software di un microcomputer, che si usa per fornire segnali di abilitazione alle operazioni di I/O di un dispositivo esterno. Pag. 17-2 e 21-2. |
| *Impulso di selezione indirizzi (Address select pulse)* | Un impulso di clock generato dal software di un microcomputer, utilizzato per fornire segnali di abilitazione per l'I/O memory mapped. Pag. 21-3. |
| *Impulsi di sincronizzazione (Syncronization pulses)* | Impulsi originati da una apparecchiatura trasmittente ed inviati in una apparecchiatura ricevente per mantenere le due apparecchiature al passo. Pag. 16-17. |
| *Indicatore a LED (LED lamp monitor)* | Diodi ad emissione di luce (LED), accesi nello stato logico 1, e spenti nello stato logico 0. Pag. A-23 |
| *Indirizzo (Address)* | Un gruppo di bit che identificano una specifica locazione di memoria o un dispositivo di I/O. Un microcomputer 8080 usa sedici bit per identificare una specifica locazione di memoria e otto bit per identificare un dispositivo di I/O. Pag. 16-12. |
| *Indirizzo di memoria (Memory address)* | Vedi indirizzo. Pagg. 2-7 e 16-12. |
| *Ingressi asincroni (Asynchronous inputs)* | Quei pin di ingresso in un flip-flop che possono influire sullo stato di uscita indipendentemente dal clock; sono indicati con il nome di preset, reset e clear. Pag. 11-5 |
| *Ingressi sincroni (Synchronous inputs)* | Quegli ingressi di un flip-flop che non controllano l'uscita direttamente, come fanno quelli di un gate, ma solo sotto controllo ed abilitazione del clock. Pagg. 11-5 e 16-17. |
| *Ingresso di clock (Clock input)* | Quel terminale di un flip-flop il cui stato, o cambiamento di stato, controlla l'ingresso di dati in un flip-flop attraverso gli ingressi sincroni, con conseguente controllo dello stato di uscita del flip-flop. Il segnale di clock attua due funzioni: (1) permette l'ingresso dati al flip-flop; (2) dopo l'ingresso, dirige concordemente il cambiamento di stato del flip-flop. Pag. 11-5 |

| | |
|---|---|
| *I/O in mappa di memoria (Memory mapped I/O)* | Un termine collegato al 6800, 8080 e ad altri microcomputer. Le istruzioni di I/O sono istruzioni con riferimento alla memoria e il trasferimento dei dati avviene, nel caso dell'8080, fra il dispositivo di I/O e uno qualunque dei registri universali all'interno del chip. Pagg. 20-2, 21-2 e 21-38. |
| *I/O in memoria (Memory I/O)* | Vedere I/O in mappa di memoria. Pag. 20-2. |
| *I/O tramite l'accumulatore (Accumulator I/O)* | Un termine associato ai sistemi a microcomputer basati sull'8080. Le istruzioni di I/O sono IN e OUT e il trasferimento dei dati avviene fra il dispositivo di I/O e l'accumulatore all'interno dell'8080. Pagg. 20-2, 20-40 e 21-2. |
| *Ingresso/uscita (Input/output)* | Termine generale per le apparecchiature usate per comunicare con un computer e con dati coinvolti nella comunicazione. Pag. 16-19 |
| *Interrogazione ciclica (Polling)* | Un test periodico dei dispositivi di ingresso/uscita o di controllo per determinare la loro condizione o stato, cioè pieno/vuoto, acceso/spento, occupato/libero, fatto/non fatto, ecc. Pag. 23-9. |
| *Interruzione (Interrupt)* | In un computer digitale, una pausa nella normale esecuzione di un programma in modo che il programma può essere ripreso più tardi da quello stesso punto. Pagg. 16-15 e 23-10. |
| *Interruzione a più livelli (Multilevel interrupt)* | Vengono fornite molte linee di interruzione indipendenti, ognuna delle quali dà luogo ad un'azione specifica. L'interrogazione ciclica non è necessaria a meno che venga eseguito un OR sui dispositivi multipli su uno degli ingressi. Pag. 23-10. |
| *Interruzione su una sola linea (Single-line interrupt)* | Un segnale d'interruzione che è inserito nel computer su di una sola linea e provoca lo svolgersi di un'azione ben definita. Viene eseguito un OR, su dispositivi multipli, su questa linea, e una routine a interrogazione ciclica deve determinare quale dispositivo ha causato l'interruzione. Pag. 23-10. |
| *Interruzione vettorizzata (Vectored interrupt)* | Ogni dispositivo punta, o vettorizza, il controllo del computer su routine di servizio specifiche per dispositivi d'interruzione. Pag. 23-12. |
| *Interruzioni con priorità (Priority interrupts)* | Interruzioni messe in ordine di importanza in modo che alcuni dispositivi con priorità hanno la precedenza sugli altri. Pag. 23-18. |
| *Invertitore (Inverter)* | Dispositivo digitale che complementa un segnale d'ingresso digitale. Pag. 7-9 |
| *Istruzione a due byte (Two byte instruction)* | Istruzione consistente in una informazione che occupa due successive locazioni di memoria. Pag. 3-2 |
| *Istruzione a tre byte (Three byte instruction)* | Istruzione consistente in una informazione che occupa tre successive locazioni di memoria. Pag. 3-2 |

| | |
|---|---|
| *Istruzione in singolo byte (Single-byte instruction)* | Istruzione consistente in 8 bit contigui, occupante una sola locazione di memoria. Pag. 3-2 |
| *Istruzione logica (Logical instruction)* | Operazione logica che è attuata da una coppia di parole dati multi-bit, in cui i bit corrispondenti di ogni parola partecipano ad un'operazione logica a due bit, come AND, OR e OR esclusivo. Pag. 8-2 |
| *Istruzione mnemonica (Mnemonic instruction)* | Istruzione di un calcolatore scritta in notazione simbolica come: ADD, SUB, MOV, DIV, MPY, STO, DIV. Pag. 2-4 |
| *Latch (Latch)* | Semplice elemento di memorizzazione. Un loop di controreazione utilizzato in un circuito digitale simmetrico, come un flip-flop, per mantenere uno stato logico. Pag. 11-5 |
| *Leggere (To read)* | Trasmettere dati da una specifica locazione di memoria a qualche altro dispositivo digitale. Un sinonimo di estrarre. Pagg. 16-15 |
| *Linguaggio (Language)* | L'insieme delle parole e dei metodi di combinazione delle parole usate da una nazione, da un popolo o da una razza. Pag. 1-2 |
| *Linguaggio mnemonico (Mnemonic language)* | Linguaggio di programmazione che è basato su simboli facilmente ricordabili e che può essere assemblato nel codice macchina di un computer. Pag. 2-4 |
| *Logica sincrona (Synchronous logic)* | Il tipo di logica digitale usata in un sistema in cui le operazioni logiche avvengono in sincronismo con gli impulsi di clock. Pag. 16-17. |
| *Loop di timing (Timing loop)* | Un loop da software che richiede un preciso periodo di tempo per essere eseguito. Pag. 17-3. |
| *Mascheramento (Masking)* | Tecnica logica in cui certi bit di una parola multi-bit sono eliminati od inibiti. Pag. 8-13 |
| *Mascheratura (Masking)* | Una tecnica in cui determinati bit di una parola vengono cancellati o inibiti. Pag. 18-45. |
| *Mascheratura esclusiva (Exclusive masking)* | Una tecnica di mascheramento in cui o si azzerano o si settano (raramente) tutti i bit su cui non si opera. Pag. 18-45. |
| *Mascheratura inclusiva (Inclusive masking)* | Una tecnica di mascheramento in cui si lasciano inalterati tutti i bit su cui non si opera. Pag. 18-45. |
| *Memoria (Memory)* | Qualunque dispositivo che può memorizzare bit di livello logico 0 o 1 in modo tale che si può accedere ed estrarre un solo gruppo di bit. Pagg. 2-6 e 16-13. |
| *Memoria a lettura/scrittura (Read/write memory)* | Memoria a semiconduttore in cui possono essere scritti e letti stati logici 0 ed 1. Pag. 2-6 |
| *Memoria a sola lettura (Read-only memory)* | Memoria a semiconduttore da cui dati digitali possono essere letti ma in cui non è possibile inserire dati. Pag. 2-7 |

| | |
|---|---|
| *Microcomputer (Microcomputer)* | Un completo computer digitale basato su un microprocessor o una famiglia di microprocessor. Pag. 2-2 |
| *Modulo (Modulo)* | Il numero dei diversi stati per cui passa un contatore, prima di ripetere da capo la sequenza. Pag. 13-2 |
| *Multiplexer (Multiplexer)* | Dispositivo digitale che può selezionare uno o un numero di ingressi e trasmettere lo stato logico di tale ingresso all'uscita. Pag. A-31 |
| *Multivibratore monostabile (Monostable multivibrator)* | Circuito digitale che ha un solo stato stabile, da cui può essere spinto a cambiare stato, ma solo per un definito intervallo di tempo, dopo di che ritorna allo stato originale. Detto anche multivibratore One-shot, single shot multivibrator, start-stop multivibrator. Pag. 15-2 |
| *Operazione (Operation)* | Specifica azione che un computer attua in base ad un'istruzione; esempio addizione, sottrazione, OR, AND etc. Pag. 3-2 |
| *Operazione mnemonica (Mnemonic operation)* | Vedere istruzione mnemonica. Pag. 2-4 |
| *Parola (Word)* | Un gruppo di bit contigui che occupano uno o più locazioni di memoria in un computer. Pagg. 2-6 e 18-11. |
| *Pilota (Driver)* | Vedere driver. Pag. 14-21 |
| *Porta (Gate)* | Vedere gate. |
| *Porta AND (AND gate)* | Circuito binario con due o più ingressi ed una sola uscita. L'uscita è allo stato logico 0 se uno qualunque degli ingressi è allo stato logico 0; l'uscita è allo stato logico 1 se tutti gli ingressi sono allo stato logico 1. Pag. 7-6 |
| *Porta NAND (NAND gate)* | Combinazione di una funzione NOT e una funzione AND in un circuito binario che ha due o più ingressi e una sola uscita. L'uscita è allo 0 logico solo se tutti gli ingressi sono all'1 logico. L'uscita è all'1 logico se uno qualunque degli ingressi è allo 0 logico. Pag. 7-8 |
| *Porta NOR (NOR gate)* | Una porta OR seguita da un invertitore; l'uscita di tale circuito binario è allo 0 logico se uno qualunque degli ingressi è all'1 logico; è all'1 logico se tutti gli ingressi sono allo 0 logico. Pag. 7-10 |
| *Porta NOT (NOT gate)* | Circuito binario con una sola uscita sempre allo stato logico opposto a quello dell'ingresso. Detto anche circuito invertitore. Pag. 7-9 |
| *Porta OR-esclusivo (Exclusive-OR gate)* | Circuito binario con due ingressi ed una sola uscita in cui l'uscita è allo stato logico 1 quando gli ingressi sono a stati logici differenti. L'uscita è a 0 se entrambi gli ingressi sono allo stesso stato logico. Pag. 7-11 |

| | |
|---|---|
| *Prelevamento dalla memoria (Fetch)* | Una delle due parti funzionali del ciclo di una istruzione. Le azioni di acquisizione di un indirizzo di memoria e poi di un'istruzione o di un byte di dati dalla memoria. Pag. 18-12 |
| *Preset (Preset)* | Ingresso asincrono utilizzato per controllare lo stato logico dell'uscita Q di un flip-flop. I segnali in ingresso attraverso il preset, determinano il passaggio dell'uscita Q verso lo stato logico 1. L'ingresso di preset non può far apparire 0 all'uscita. Pag. 11-17 |
| *Programma per computer (Computer program)* | Una sequenza di istruzioni che, prese come gruppo, permettono al computer di effettuare un dato compito. Pagg. 2-2 e 5-2 |
| *PSW (PSW)* | Abbreviazione di "parola di stato del processore". I contenuti dell'accumulatore e i cinque flag di stato nel microprocessore 8080. Pag. 18-61. |
| *Puntatore dello stack (Stack pointer)* | Il registro a 16 bit nel microprocessore 8080 che memorizza l'indirizzo di memoria del top dello stack, una regione di memoria di lettura/scrittura che memorizza informazioni temporanee. Pag. 18-11. |
| *Radice (Radix)* | Vedere base. Pag. 1-4 |
| *Registro (Register)* | Un circuito di memorizzazione elettronica digitale a breve termine la cui capacità è di solito di una parola. Pagg. 3-7 e 18-8. |
| *Registro di istruzione d'interruzione (Interrupt instruction register)* | Un registro esterno a 8 bit che permette a un'istruzione di venire forzata nel registro istruzione all'interno dell'8080 durante un'interruzione. Pag. 23-49. |
| *Registro istruzioni (Instruction register)* | Il registro a 8 bit nel microprocessore 8080 che memorizza il codice istruzione dell'istruzione che viene eseguita in quel momento. Pag. 18-11. |
| *Registro non specializzato (General purpose register)* | In un microcomputer basato sull'8080, vi sono 6 registri ad 8 bit, utilizzati per memorizzare informazioni in modo temporaneo. I registri sono: B, C, D, E, H, L. Pag. 6-2 |
| *Registro universale (General purpose register)* | Nel microcomputer 8080, i registri a 8 bit che possono prendere parte ad operazioni logiche od aritmetiche con i contenuti dell'accumulatore. Pag. 18-8. |
| *Reset (Reset)* | Ingresso asincrono utilizzato per controllare lo stato logico dell'uscita Q di un flip-flop. I segnali in ingresso attraversano il reset, determinano il passaggio dell'uscita Q verso lo stato logico 0. L'ingresso di reset non può far apparire 1 all'uscita Q. Pag. 11-7 |

| | |
|---|---|
| *Resistenza (Resistance)* | Proprietà dei conduttori che in funzione della loro dimensione, materiale e temperatura, determina la corrente prodotta da una data differenza di potenziale. Quella proprietà di una sostanza che ostacola il passaggio della corrente con conseguente dissipazione di potenza sotto forma di calore. L'unità di misura della resistenza è l'ohm. Quest'ultimo è definito come la resistenza attraverso cui una differenza di potenziale di un volt produce una corrente di un ampére. Pag. 9-11 |
| *Resistore (Resistor)* | Componente elettronico che connesso in un circuito elettrico introduce una specifica resistenza. Pag. 9-11 |
| *Routine di servizio (Service routine)* | Una subroutine del computer che serve un dispositivo d'interruzione. Pag. 23-20. |
| *Schema elettrico (Schematic diagram)* | Rappresentazione grafica di circuiti elettronici. Pag. 9-10 |
| *Scrivere (To write)* | Trasmettere dati da qualche altro dispositivo digitale ad una specifica locazione di memoria. Un sinonimo di memorizzazione. Pag. 16-15. |
| *Segnale binario (Binary signal)* | Tipicamente una corrente od una tensione che fornisce una informazione in termini di cambiamento tra due differenti stati: stato logico 0 e stato logico 1. Pag. 14-2 |
| *Segnale di gate (Gate signal)* | Vedere impulso di porta. Segnale che permette ad un circuito di porta (gate circuit) di far passare un segnale. Pag. 14-20 |
| *Segnali digitali (Digital signals)* | Segnali discreti o discontinui i cui vari stati sono intervalli discreti. Pag. 14-2 |
| *Set (Set)* | Vedere preset. Pag. 11-7 |
| *Simbolo (Symbol)* | Carattere scritto utilizzato per rappresentare qualunque cosa: lettere, figure, segni convenzionali indicanti un qualche oggetto, processi, etc.. Pag. 9-10 |
| *Simbolo Booleano (Boolean symbol)* | Simbolo utilizzato per rappresentare una specifica operazione booleana. Pag. 8-2 |
| *Simbolo mnemonico (Mnemonic symbol)* | Un simbolo scelto per aiutare la memoria umana: es. MPY è la abbreviazione di "multiply". Pag. 2-4 |
| *Sincronizzare (To synchronize)* | Mettere un elemento di un sistema al passo con un altro. Pag. 16-17. |
| *Sincrono (Synchronous)* | In passo o in fase, applicato a due dispositivi o macchine. Un termine applicato ad un computer, in cui l'esecuzione di una sequenza di operazioni è controllata da segnali di clock o da impulsi di clock. Nello stesso tempo. Pag. 16-17. |

*Sistema di acquisizione dati (Data logger)* — Uno strumento che scandisce automaticamente i dati prodotti da un'altro strumento o processo, e registra le letture di dati per uso successivo. Pag. 22-2.

*Software (Software)* — La totalità dei programmi e delle routine usate per estendere la capacità dei computer, compresi compilatori, assemblatori, linker-loaders, traduttori e subroutine. Opposto di hardware. Pag. 16-7.

*Switch (Switch)* — Dispositivo elettrico o meccanico che attiva o interrompe una linea interessata al passaggio di corrente, o che indirizza la corrente a differenti possibili vie. Pag. 14-21

*Switch logico (Logical switch)* — Dispositivo meccanico che presenta sia uno stato logico 0 che uno stato logico 1 al terminale di uscita. Pag. 9-12

*Sync* — Abbreviazione di sincrono, sincronizzazione, ecc. Pag. 16-17.

*Tabella della verità (Truth table)* — Tabulazione che mostra le relazioni tra tutti i livelli logici di uscita di un circuito digitale in funzione di tutte le possibili combinazioni dei livelli logici in ingresso; da cui la completa caratterizzazione funzionale di un dato circuito. Pag. 7-3

*Tempo di discesa (Fall time)* — Il tempo necessario al fronte di discesa di un impulso per passare dal 90% al 10% del suo valore iniziale. In elettronica digitale, è il tempo necessario ad una tensione di uscita di un circuito digitale per cambiare da alto livello (1 logico) a basso livello di tensione (0 logico). Pag. 11-10

*Tempo di salita (Rise-time)* — Il tempo richiesto dal fronte di salita di un impulso positivo per passare dal 10% al 90% del valore finale. È proporzionale alla costante di tempo ed è la misura della pendenza del fronte d'onda. In elettronica digitale, il tempo richiesto perchè una tensione in uscita da un circuito digitale cambi da basso livello (0 logico) ad alto livello (1 logico). Pag. 11-10

*Tempo totale di ritardo di propagazione (Propagation delay)* — Detto anche delay di propagazione.
È il tempo richiesto da un segnale logico per passare attraverso una serie di dispositivi logici. Vi sono 4 tipi di ritardi circuitali: memorizzazione (storage) salita (rise), discesa (fall) e ritardo di turn-on. Tempo di ritardo totale tra quando un segnale d'ingresso attraversa il punto di soglia di un valore di tensione e quando la corrispondente uscita attraversa lo stesso punto di tensione. Pag. 11-10

*Teorema di DeMorgan (DeMorgan's theorem)* — Teorema secondo cui l'inversione di una serie di AND è uguale alla stessa serie di OR singolarmente invertiti; oppure l'inversione di una serie di OR è uguale alla stessa serie di AND singolarmente invertiti; simbolicamente è:

$$A \cdot B \cdot C = A + B + C$$
$$A + B + C = A \cdot B \cdot C$$

*To gate (To gate)* — Controllare il passaggio di un segnale digitale attraverso un circuito digitale. Pag. 15-15

| | |
|---|---|
| *To strobe* <br> *(To strobe)* | Attivare un circuito digitale. <br> Vedere abilitare (enable). Pagg. 11-12 e 14-16 |
| *To trigger* <br> *(To trigger)* | Fornire un impulso che fa partire una certa azione. <br> Può anche essere il fronte di un impulso. Vedere abilitare (enable). Pagg. 11-12, 13-6 e 14-21 |
| *Uscita a collettore aperto* <br> *(Open collector output)* | Un'uscita di un dispositivo a circuito integrato in cui il resistore finale di pull-up nel transistor di uscita per il dispositivo è mancante e deve essere fornito dall'utente in modo che il circuito sia completo. Pag. 17-12. |
| *XOR* <br> *(XOR)* | Vedere porta OR-esclusivo. Pag. 7-11 |

# APPENDICE 3

# OUTBOARDS®

*Outboard®* è un marchio registrato della E-L Instruments, Inc. e indica un elemento, che svolge una funzione ausiliaria, montato su un piccolo circuito stampato adatto al collegamento diretto sulla piastra SK-10. I collegamenti +5 V e massa per gli Outboards sono realizzati attraverso le file di terminali senza saldature presenti sull'SK-10. Sono ottenuti dalle due file dei terminali. I pin di ingresso-uscita vengono elettricamente collegati ai sets di 5 terminali senza saldatura presenti nella parte interna della piastra SK-10. Le caratteristiche fisiche di un Outboard possono essere facilmente dedotte dalla figura A1, che rappresenta l'Outboard LR-7, dual pulser visto da sotto. I collegamenti con l'SK-10 avvengono in 6 punti: +5V (fila esterna del bus di terminali di alimentatione), 0V (parte interna del bus di terminali di alimentazione) e 4 terminali di ingresso. Per realizzare queste connessioni, dovete premere leggermente l'Outboard sulla piastra SK-10 nell'opportuna posizione.
I vantaggi del concetto di Outboard possono essere così riassunti: gli Outboard sono compatti, facilmente reperibili, economici, portatili e possono essere collocati ovunque sulla piastra SK-10. I collegamenti +5V e Massa per gli Outboards sono realizzati attraverso le due file di terminali senza saldature presenti sull'SK-10.

Gli Outboards possono essere raggruppati in 7 categorie differenti:

| | |
|---|---|
| Outboards di display: | LR-6 Lamp monitor Outboard<br>LR-4 Seven-segment display Outboard<br>LR-26 Latch/display Outboard<br>LR-27 Octal latch/display Outboard<br>LR-28 Three digit lach/display Outboard |
| Outboards di ingresso digitale: | LR-2 Logic switch Outboard<br>LR-7 Dual pulser Outboard<br>LR-5 Clock Outboard<br>LR-20 Monostable Outboard<br>LR-10 Programmable timer Outboard |
| Outboards di comunicazione: | LR-21 UART Outboard<br>LR-14 TTL/20mA current loop interface Outboard<br>LR-13 TTL/RS-232C interface Outboard |
| Outboards di funzioni digitali: | LR-19 Lach Outboard<br>LR-22 Decoder Outboard<br>LR-17 Decade counter Outboard<br>LR-18 Binary counter Outboard<br>LR-23 Multiplexer Outboard<br>LR-23 Multiplexer Outboard<br>LR-12 Driver/inverter/NOR Outboard. |

Oltre a questi vi è l'LR-1 Power Outboard (Outboard di alimentazione) e l'LR-25 breadboarding station Outboard.

## POWER OUTBOARDS

L'alimentazione è applicata alla piastra SK-10 tramite l'*Outboard di alimentazione (Power Outboard)* LR-1 (figura A-2) o l'LR-25 Breadboarding Station Outboard mostrato nella figura A-3.

*Figura A-1.* Outboard LR-7 visto da sotto. Si notano i due pin di alimentazione e i 4 pin di ingresso-uscita.

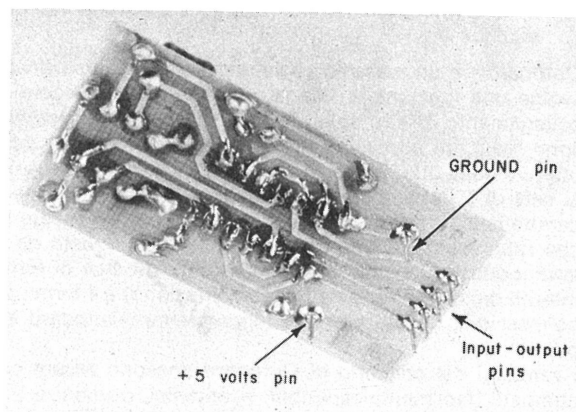



*Figura A-2.* L'LR-1 Power Outboard (Outboard di alimentazione)

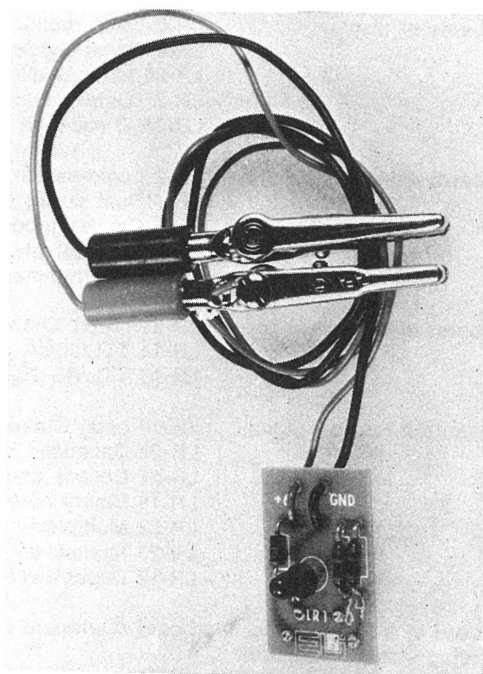



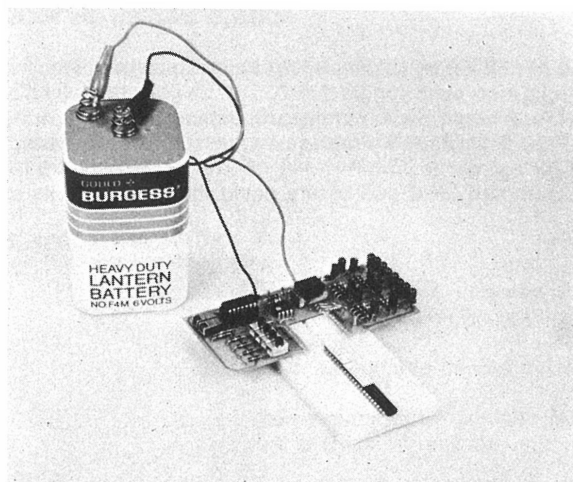
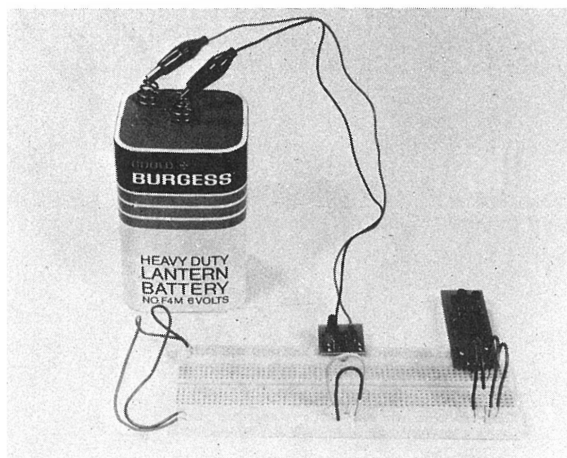
*Figura A-3.* L'LR-25 Breadboarding Station Outboard montato su una piastra SK-50 e alimentato da una batteria a 6V.

La circuiteria di alimentazione di questi due Outboard controlla la tensione c.c. applicata all'SK-10.
I LED indicano quando è applicata una alimentazione c.c. corretta, quando non è corretta, quando le prese a coccodrillo sono state connesse erroneamente alla batteria o ai terminali dell'alimentatore (i LED sono spenti) e, per ultimo, quando la tensione della batteria è bassa (i LED sono semiaccesi).
Un diodo raddrizzatore previene dannose conseguenze risultanti da errata connessione delle prese a coccodrillo ai terminali della batteria. Se tali prese sono collegate in modo errato, non viene applicata alcuna tensione al breadboard ed i LED restano spenti.
Nella figura A-4 è indicato un esempio di collegamento

*Figura A-4.* Esempio di utilizzo di una batteria per alimentare l'SK-10. L'LR-1 Power Outboard serve per dare tensione a 50 terminali del bus di alimentazione. I restanti 50 terminali sono collegati ai precedenti con ponticelli, che devono sempre restare collegati.

## LOGIC SWITCH OUTBOARDS

Lo *switch logico (logic switch)* è un dispositivo meccanico che presenta ai suoi terminali di uscita uno stato logico 1 o 0.
L'LR- 2 Logic Switch Outboard, indicato nella figura A-5, e anche presente sull'Outboard LR-25 di figura A-3. Fornisce 4 switch logici che possono commutare tra 0 V (0 logico) e +5 V (1 logico).
L'Outboard LR-2 può essere posto indifferentemente sui due lati del breadboard.

*Figura A-5.* L'LR-2 Logic Switch Outboard

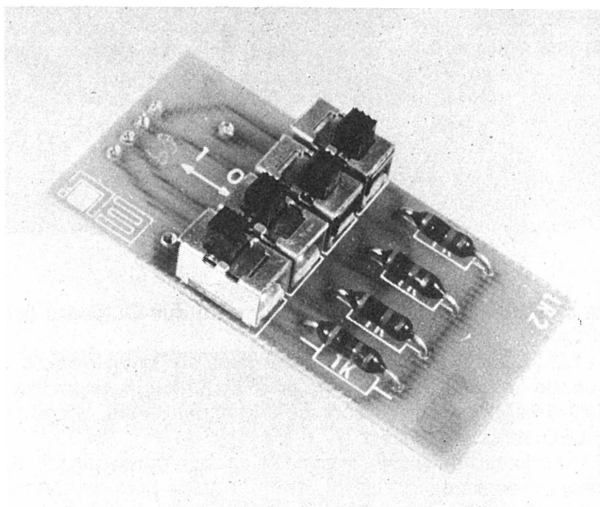
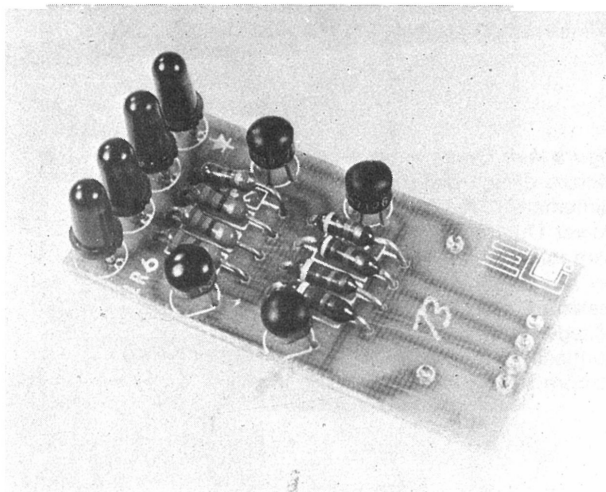



*Figura A-6.* L'LR-6 LED Lamp Monitor Outboard

## LED LAMP MONITOR OUTBOARDS

Un *indicatore a LED (led lamp monitor)* è di fatto un diodo ad emissione di luce (LED) che è acceso nello stato logico 1 e spento nello stato logico 0.
È utilizzato come display per segnali digitali binari.
L'LR-6 LED Lamp Monitor Outboard contiene 4 LED, mentre l'LR-25 ne contiene 8.
L'utilizzo di transistor per pilotare il LED, riduce la corrente per ogni LED a soli 1,5 mA.

## PULSER OUTBOARDS

Un *generatore di impulsi (pulser)* è uno switch logico che genera un singolo impulso di clock, che è un ciclo logico completo da 0 logico ad 1 logico ed ancora a 0 logico oppure da 1 a 0 logico ed ancora ad 1 logico.
Per avere un generatore di impulsi, occorre eliminare i *rimbalzi da contatto* causati dagli switch meccanici utilizzati.
Un rimbalzo da contatto (contact bounce), il fermarsi e l'interrompersi di un contatto in modo incontrollato quando i contatti dello switch sono aperti o chiusi, è un fatto abbastanza comune.
In alcune applicazioni digitali, i rimbalzi non sono un problema. In molti circuiti digitali, tuttavia, è necessario che l'uscita di un generatore di impulsi sia esente da rimbalzi, per permetterne l'uso in circuiti digitali *clocked,* come quelli che richiedono un opportuno · tempo di clock per le loro operazioni.
L'LR-7 Dual Pulser Outboard, di cui è mostrata una vista dall'alto in figura A-7 e dal basso in figura A-1 utilizza 4 porte NAND a 2 ingressi in un singolo chip di circuito integrato, 7400 per produrre una coppia di *debounced pulsers,* la cui uscita è esente da rimbalzi di contatto. Sono presenti le uscite complementari relative agli stati logici 0 ed 1, dette "0" ed "1", per ogni generatore di impulsi sull'LR-7 ed anche sull'LR-25 (figura A-3).
Per attivare ciascun generatore di impulsi, occorre premere il pulsante di plastica.
L'operazione può essere fatta con la punta di una penna a sfera.

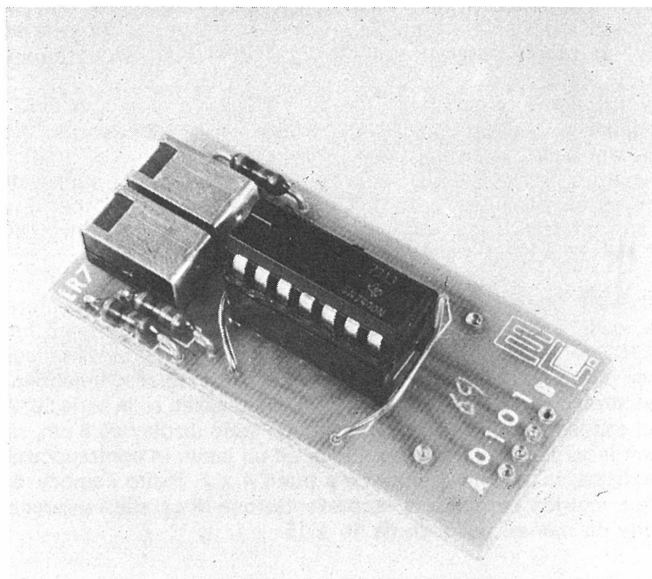



*Figura A-7.* L'LR-7 Dual Pulser Outboard.

**DISPLAY E LATCH/DISPLAY OUTBOARDS**

Un *display* è un dispositivo che fornisce la visualizzazione di un segnale elettronico.Un indicatore a LED è un display per un singolo bit di informazione binaria. Quattro LED possono mostrare 4 bit di informazione binaria.

Nell'elettronica digitale, vengono richieste le visualizzazioni delle informazioni:

- Singolo bit

- Gli stati logici di singolo flag, uscite dalle porte, uscite dai flip-flop

- Quattro bit

  Codice binario decimale codificato (BCD), codice esadecimale, ed altri codici binari a 4 bit, come quelli relativi ai microprocessor a 4 bit.

- Codice ottale ad 8 bit

  Codici di istruzione espressi su 8 bit, codici di dispositivo, byte di indirizzo di memoria, byte di dati; tutti utilizzati in microprocessor ad 8 bit come l'MMD-1. Codici ad 8 bit come l'ASCII e l'EBCDIC

- Multipli di 4 bit

  La visualizzazione di numeri decimali in codice BCD nei contatori, misuratori di frequenza, multimetri digitali, pannelli di misura, ed altra strumentazione digitale.

Sono disponibili Outboard per ciascuno di questi tipi di informazioni.

Per visualizzazione di singoli bit, si usa l'LR-6. Per i 4 bit l'LR-6, l'LR-4 Seven-Segment LED Display Outboard (figura A-8) oppure l'LR-26 Latch/Display Outboard (figura A-9). L'Outboard a sette segmenti contiene un display in cui sette segmenti sono posti in modo tale da rappresentare i digit da 0 a 9 attraverso l'accensione selettiva di certi segmenti.

Sull'LR-4 vi è un circuito integrato 7447 decoder/driver, un 5082-7730 della Hewlett-Packard oppure un display numerico Opcoa SLA-1, e sette *resistori limitatori di corrente* per salvaguardare il display.
I quattro ingressi ABCD verso l'Outboard generano i numeri da 0 a 9, cinque simboli ed il blank: quindi in totale 16 diversi stati sono disponibili per il display. Sull'Outboard LR-4 sono presenti 3 ingressi addizionali al chip 7447, il BLANKING INPUT (I), il BLANKING OUTPUT (O) ed il LAMP TEST (T).

L'LR-26 Single Latch/Display Outboard contiene un indicatore numerico singolo od esadecimale della HP. Cinque pin di ingresso all'Outboard funzionano come i 4 ingressi ABCD in codice binario decimale codificato o ingressi esadecimali, più uno strobe (STB) che permette di effettuare un *latch* di 4 bit in ingresso indefinitamente. Tutti gli Outboard di latch/display, l'LR-27, LR-28, LR-26, sono basati sulla serie 5082-7300 di indicatori numerici ed esadecimali della HP. Gli indicatori sono display ad 8 pin, sia decimali che esadecimali, con incorporato un decoder/driver ed un latch. In contrapposizione all'LR-4, i display sono realizzati in termini di matrice a punti 4 x 7, molto comoda da leggere.
Tale matrice permette la rappresentazione di caratteri esadecimali come A, B, C, D, E, F, cioè da numeri decimali da 10 a 15.

*Figura A-8.* L'LR-4 Seven-Segment LED Display Outboard.
In questo Outboard, la notazione di ingresso 1248 sostituisce la notazione ABCD. La corrispondenza tra i due tipi di notazione è: A=1. B=2, C=4, D=8.

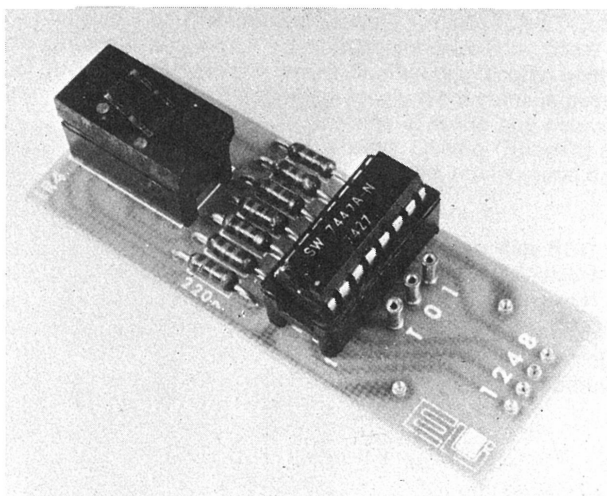



*Figura A-9.* L'LR-26 Single Latch-Display Outboard

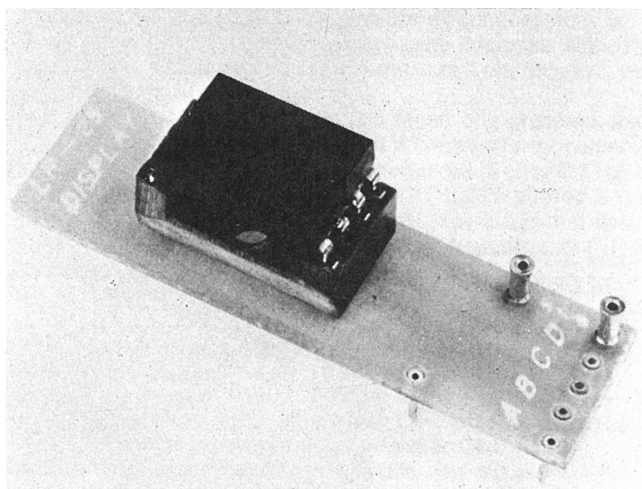



*Figura A-10.* L'LR-27
Octal Latch/Display
Outboard

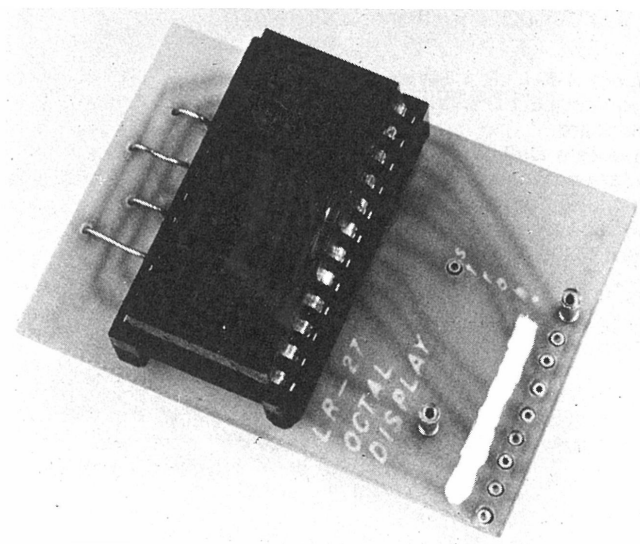
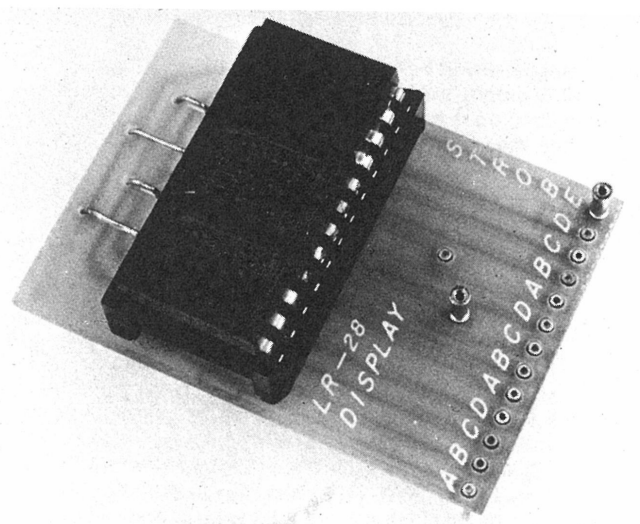



*Figura A-11.* L'LR-28
Three Digit Latch/Display
Outboard

Per la visualizzazione del codice ottale ad 8 bit, discusso nel capitolo 1, si usa l'LR-27 Octal Latch/Display Outboard. Questo Outboard contiene 3 set di indicatori numerici, come si vede in figura A-10, ciascuno dei quali ha l'ingresso D a massa. Due set di indicatori hanno gli ingressi A,B e C, mentre il terzo set ha l'ingresso C a massa e solo A e B come ingressi. Gli ingressi di Strobe (STB) da 3 indicatori sono collegati insieme in modo che risulta possibile memorizzare un'intera parola ad 8 bit espressa su 3 digit. Questo Outboard è utilizzato come *bus monitor* nell'MMD-1 e permette di verificare molti dei trasferimenti dati che avvengono sul bus ad 8 bit del microprocessor 8080A.

Per la visualizzazione di molti digit decimali, ciasuno dei quali è codificato in codice BCD, si usa l'LR-28 Three Digit Latch/Display Outboard. Questo Outboard contiene 3 indicatori numerici od esadecimali e 3 set di ingressi ABCD per ciascun indicatore. Gli ingressi di strobe a tutti e tre gli indicatori sono collegati insieme permettendo così di memorizzare numeri a 3 decimali. Una coppia di indicatori su tale Outboard può essere usata per controllare un byte ad 8 bit di un microprocessor, in termini di due digit esadecimali. Questo Outboard è indicato in figura A-11.

## CLOCK OUTBOARDS

Per *clock* si intende un qualunque dispositivo che genera un *impulso di clock,* ciclo logico completo (con questo intendo un segnale che effettua una transizione dallo stato logico 0 allo stato logico 1 e ancora allo stato logico 0 (oppure una transizione dallo stato logico 1 allo stato logico 0 e ancora allo stato logico 1).
L'LR-5 Clock Outboard (figura A-12), presente anche sull'Outboard LR-25 (figura A-3), genera una sequenza di impulsi di clock detta anche *treno di impulsi di clock,* la cui frequenza è funzione della capacità di timing inserita tra due pin visibili sull'Outboard. Tali pin sono facilmente identificabili sull'LR-5 e sull'LR-25. In assenza di condensatori, la frequenza del clock è circa 90 KHz. I condensatori che possono essere impiegati vanno da 10 pF a 100 $\mu$F. Nel caso di condensatori elettrolitici, occore identificare il pin negativo; per l'LR-5 è il pin direttamente al di sotto delle lettere LR-5.
Il cuore di questo Outboard di clock è il circuito integrato 555 timer, che presenta una stabilità di frequenza di circa 0,1%. Sull'LR-5, un LED visualizza il corretto funzionamento del clock. Un condensatore di 0,33 $\mu$F consente di ottenere una frequenza di circa 0,7 Hz; la vostra frequenza di clock per una capacità identica può mutare del $\pm$ 20% attorno a tale valore. Il pin di uscita dell'Outboard è indicato con CLK. Possono essere aggiunti dei resistori esterni ai restanti ingressi per aumentare il limite superiore di frequenza dell'LR-5. Il pin di uscita dell'LR-25 è indicato con CK, come mostrato in figura A-3.

## BREADBOARDING STATION OUTBOARD

L'LR-25 Breadboarding Station Outboard, indicato sia nella figura A-3 che nella figura A-13, comprende i seguenti Outboards: LR-1, LR-2. LR-5, LR-6 (due) LR-7 e LR-26. Questo Outboard viene utilizzato come elemento di breadboarding completo. Con questo Outboard, è possibile attuare qualunque esperimento che richiede non più di 4 switch logici, due generatori di impulsi con logica antirimbalzo, un clock, otto LED, ed un latch/display collegato in parallelo a 4 indicatori a LED;

*Figura A-12.* L'LR-5 Clock Outboard. La capacità di timing è inserita tra i due pin fra cui vi è la lettera C.

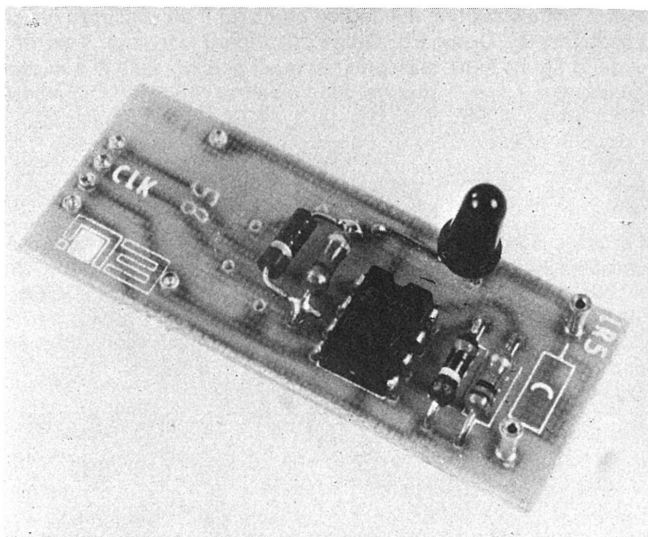
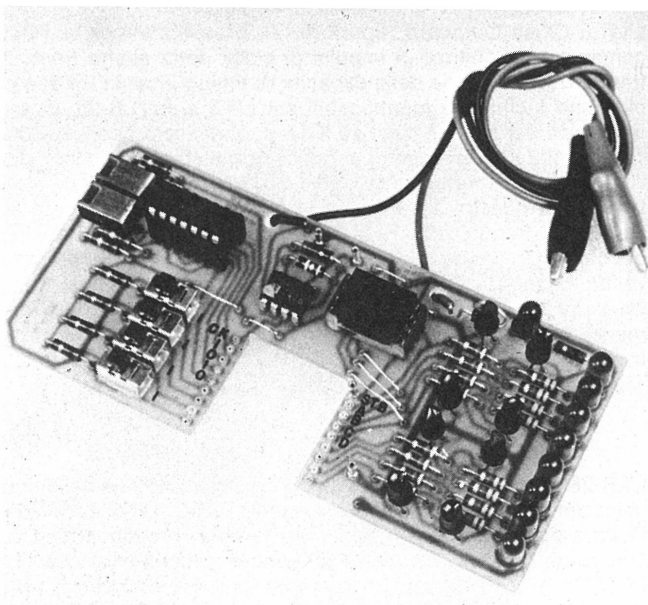



*Figura A-13.* L'LR-25 Breadboarding Station Outboard.

## DECODER OUTBOARD

In uno dei capitoli del Bugbook VI imparerete come decodificare il *"device code" (codice di dispositivo)* di un microprocessor per produrre degli individuali *"device select pulses" (impulsi di selezione di dispositivo)* utilizzabili per abilitare o disabilitare dispositivi digitali elettronici. L'LR-22 Decoder Outboard, mostrato in figura A-14, contiene un 74154, decodificatore singolo chip da 4 linee a 16, e un contenitore DIP a 16 pin. Questo Outboard può essere usato per estendere le 5 uscite di selezione dispositivo presenti sul microcomputer MMD-1 (nel blocco di decodifica I/0). Ogni LR-22 può generare 16 differenti "impulsi di selezione di dispositivo".

## MONOSTABLE OUTBOARD

Un *multivibratore monostabile* è un circuito digitale che ha un solo stato stabile, da cui con un impulso di sincronizzazione (trigger) può essere spinto (con un impulso di sincronizzazione-trigger) a cambiare stato, ma solo per un predeterminato intervallo di tempo, al termine del quale ritorna nello stato originario. Tale circuito può essere usato per produrre dei clock singoli di durata definita. L'LR-20 monostable Outboard (figura A-15) contiene un chip multivibratore monostabile retriggerabile 74122 che genera singoli impulsi di clock con l'ausilio di un piccolo potenziometro da 25 kΩ e di una capacità esterna. La durata degli impulsi va da 70 ns a 5 ms.

## LATCH OUTBOARD

Un *latch* è un elemento di memorizzazione di una informazione binaria. Un singolo latch memorizza un bit di informazione. L'LR-19 Latch Outboard (figura A-16) contiene un chip 74175 positive-edge-triggered latch. Questo chip è una memoria a 4 bit con ingresso di STROBE o di CLEAR, e due uscite complementari, Q e $\overline{Q}$ per ciascun ingresso. L'Outboard LR-19 è utile per immagazzinare quattro bit di informazione dal microcomputer MMD-1. Un impulso di selezione di dispositivo è applicato all'ingresso di STROBE per acquisire e memorizzare il dato.

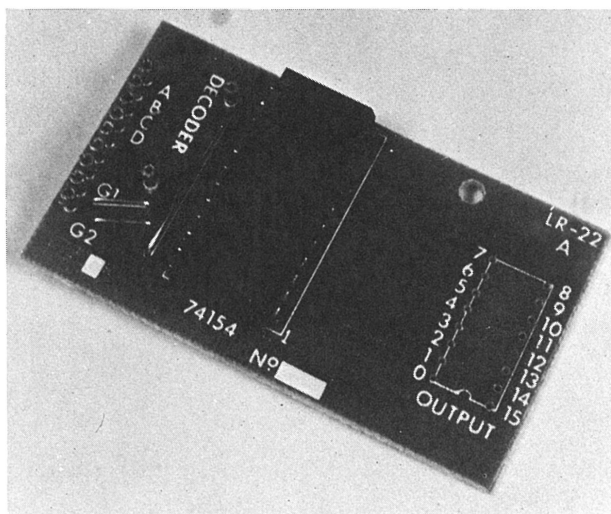



*Figura A-14.* L'LR-22 Decoder Outboard.

*Figura A-15.* L'LR-20
Monostable Outboard

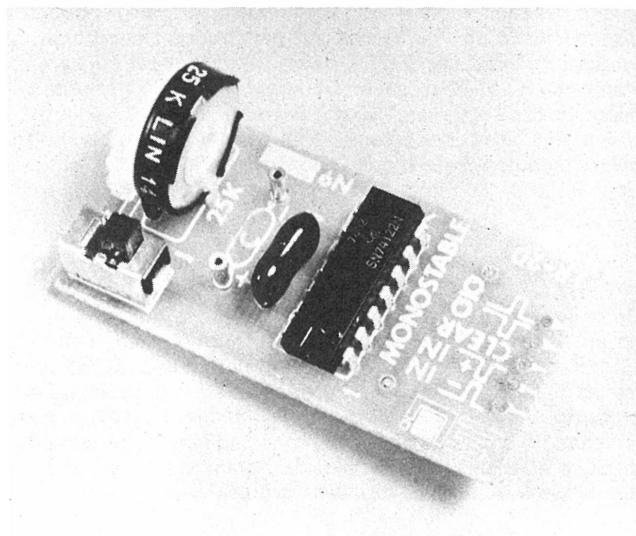
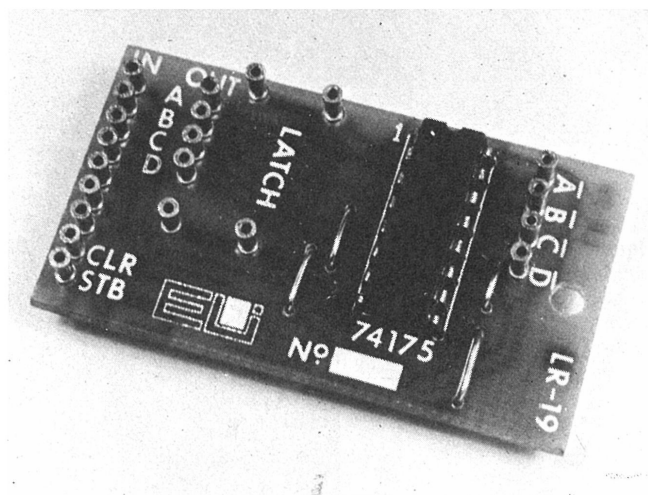



*Figura A-16.* L'LR-19
Latch Outboard

## MULTIPLEXER OUTBOARD

Un *multiplexer* è un dispositivo digitale che può selezionare uno tra più ingressi e trasferire lo stato logico di tale ingresso sull'unica uscita. Questo dispositivo agisce come unidirectional single-pole multiposition switch che passa l'informazione di un ingresso verso un'uscita. L'LR-23 Multiplexer Outboard contiene un chip 74150 da 16 linee ad 1. L'Outboard è mostrato in figura A-17.

## COUNTER OUTBOARDS

L'LR-17 Decade Counter e l'LR-18 Binary Counter Outboards sono basati, rispettivamente, sui popolari chip 7490 e 7493.
Un *contatore decimale* (decade counter) è un dispositivo logico che ha dieci stati stabili e può passare sequenzialmente tra questi stati con l'applicazione di dieci impulsi di clock. Un contatore decimale usualmente conta in sequenza binaria da 0 a 9 ricominciando poi ancora da 0. *Un contatore binario* a 4 bit (binary counter) è un dispositivo logico che ha 16 stati stabili e può ciclare tra tali stati con l'applicazione di 16 clock. Ogni contatore, come da figura A-18, contiene uno switch DPDT che permette un conteggio "free running" o che può essere resettato a 0 tramite un segnale applicato a un preciso piedino d'ingresso. Anche l'LR-17 Decade Counter Outboard può essere resettato a 9 attraverso un segnale esterno.

## DRIVER/INVERTER/NOR OUTBOARD

Sull'LR-12 Driver/Inverter/NOR/Outboard (figura A-19), un invertitore a *collettore aperto (open collector)* 7405 è utilizzato per generare una porta NOR a due ingressi, due driver e due invertitori. Dato che le uscite sono open collector, possono essere collegate insieme in wired-OR per produrre una porta AND a due ingressi e un addizionale porta NOR a due ingressi.

*Figura A-17.* L'LR-23 Multiplexer Outboard

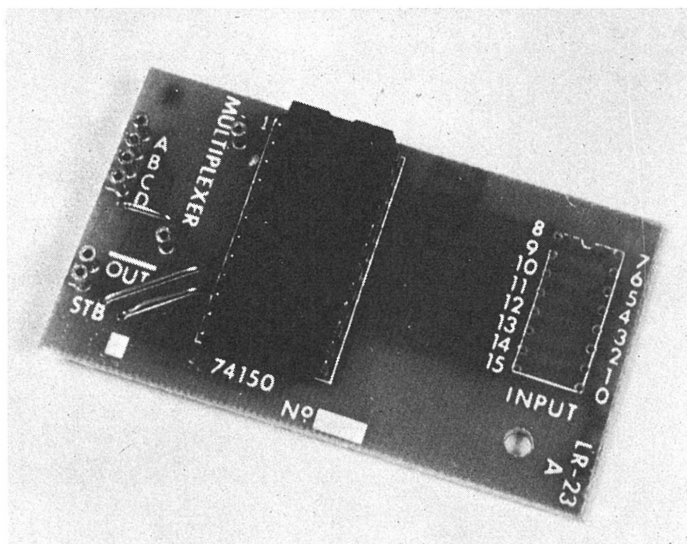



*Figura A-18.* L'LR-17/18 BCD/Binary Counter Outboard. Il tipo di Outboard dipende dal fatto che sia utilizzato il chip 7490 oppure il chip 7493.

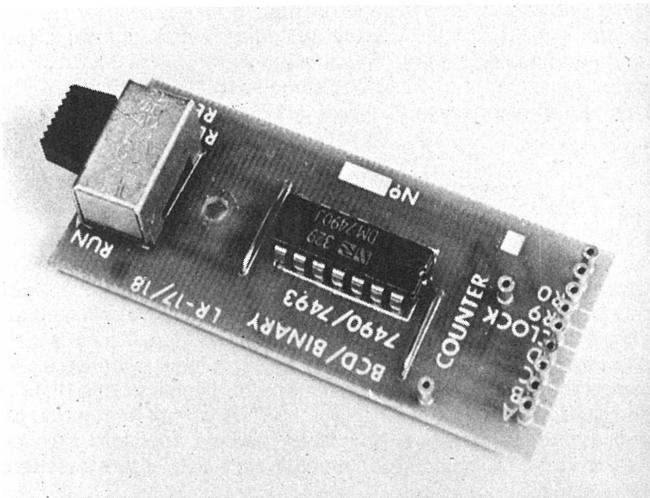
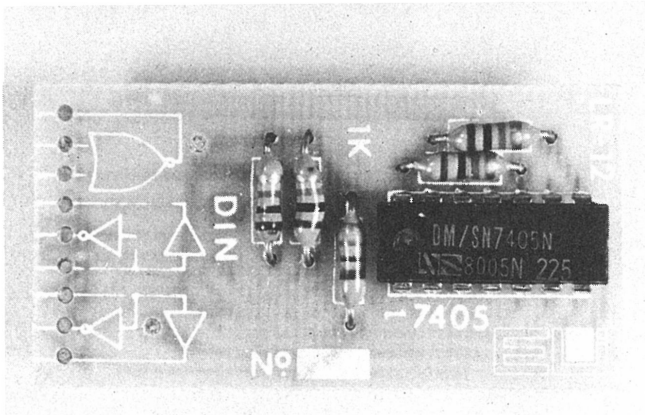



*Figura A-19.* L'LR-12 Driver/Inverter/NOR Outboard.

## UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER OUTBOARD

Se volete trasmettere dati dall'MMD-1 verso una teletype od un display a raggi catodici, dovete convertire i dati in uscita espressi su 8 bit in codice seriale asincrono ASCII. Come fare ciò è indicato nel Bugbook II, disponibile nella versione Italiana presso la JACKSON Editrice.
L'Outboard utilizzato è l'LR-21 UART, che contiene un chip UART, un "microswitch programming plug" e due contenitori DIP a 16 pin.
L'LR-21 è mostrato nella figura A-20. L'UART può trasmettere dati con una frequenza superiore a 30.000 bits.

## TTL/20ma CURRENT LOOP INTERFACE OUTBOARD

Una teletype può trasmettere o ricevere un codice ASCII solo tramite un *"20 mA current Loop"*, descritto nel Bugbook II.
L'LR-14 TTL/20 mA Current Loop Interface Outboard contiene una coppia di GE 4N35 opto-isolatori e un regolatore di corrente di 20 mA on-board. Questo Outboard interfaccia un UART ad un qualunque dispositivo asincrono che opera, relativamente alla trasmissione dati, a frequenze anche superiori a 30.000 bit/s.

## TTL/RS-232C INTERFACE OUTBOARD

Desiderando trasmettere o ricevere dati in codice ASCII tramite un *modem* collegato ad una linea telefonica, è necessario convertire i segnali TTL in segnali digitali RS-232C. Questo è possibile con l'LR-13 Line Driver/Receiver e TTL/RS-232C Interface Outboard (figura A-22).
L'Outboard contiene due 8T15 Signetics dual line driver ed un 8T16 dual line receiver, più due diodi zener che assicurano che solo la tensione ±12 V sia applicata al driver.

*Figura A-20.* L'LR-21 UART Outboard

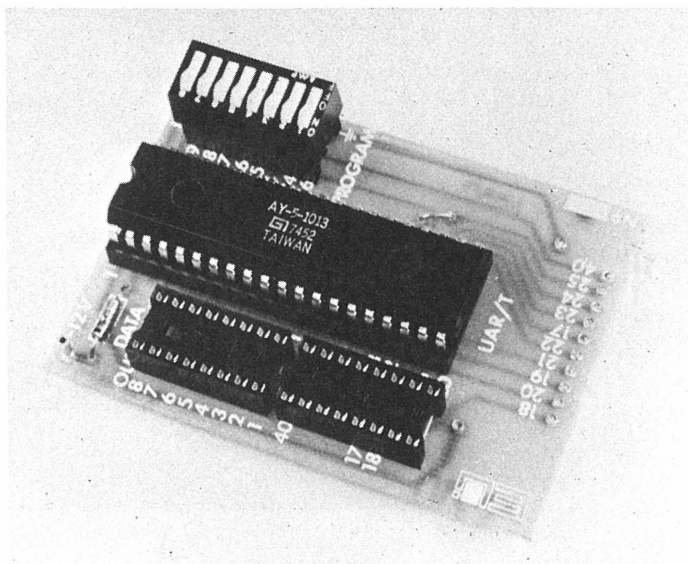



*Figura A-21.* L'LR-14 TTL/20 mA Current Loop Interface Outboard.

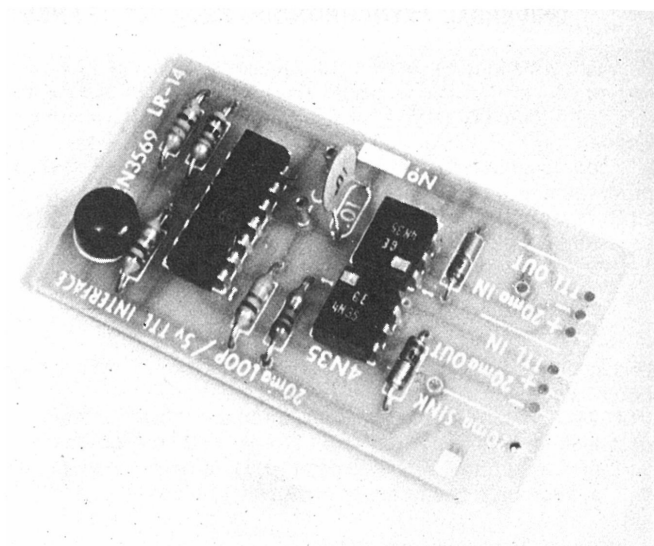
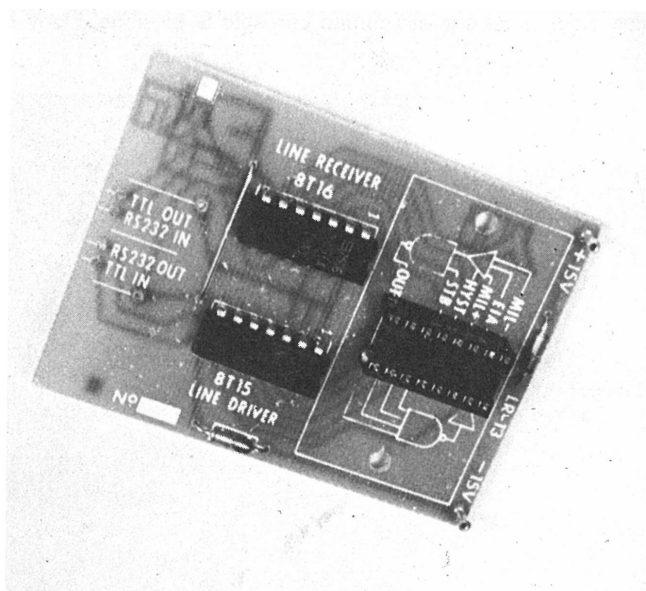
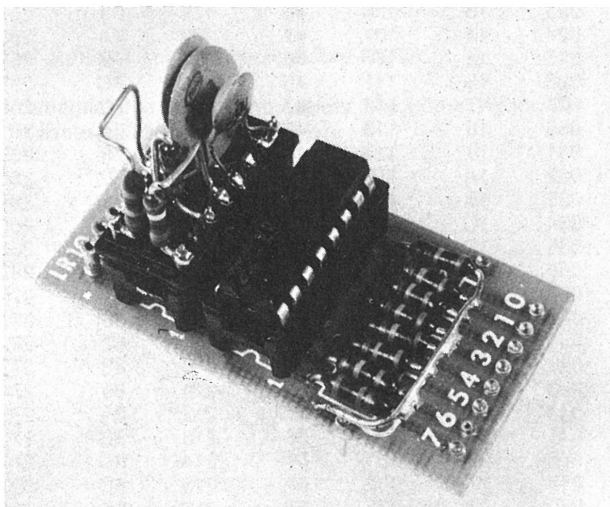



*Figura A-22.* L'LR-13 Line Driver/Receiver e TTL/RS-232. Interface Outboard

## PROGRAMMABLE TIMER OUTBOARD

L'LR-10 Programmable Timer Outboard è basato su un chip XR-2240/2340, timer/counter programmabile, che si presenta in un DIP a 16 pin. Un contenitore programmabile, indicato in figura A-23, programma il chip come timer, circuito di delay, multivibratore monostabile, ecc. Il circuito nella figura è per un semplice clock programmabile, in cui le frequenze sotto riportate possono essere prodotte come multiple della fondamentale, o più bassa, frequenza presente alla uscita del pin 0:

| Pin di uscita | Multipli della frequenza fondamentale |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |

Si noti che: $2^7=128$, $2^6=64$, $2^5=32$, $2^4=16$, $2^3=8$, $2^2=4$, $2^1=2$, $2^0=1$

*Figura A-23.* L'LR-10 Programmable Timer Outboard.

# APPENDICE 4
# TABELLA DI CONVERSIONE FRA CODICI OTTALI E CODICI ESADECIMALI

| Ottale | Esadecimale | Ottale | Esadecimale | Ottale | Esadecimale | Ottale | Esadecimale | Ottale | Esadecimale |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 00 | 063 | 33 | 146 | 66 | 231 | 99 | 314 | CC |
| 001 | 01 | 064 | 34 | 147 | 67 | 232 | 9A | 315 | CD |
| 002 | 02 | 065 | 35 | 150 | 68 | 233 | 9B | 316 | CE |
| 003 | 03 | 066 | 36 | 151 | 69 | 234 | 9C | 317 | CF |
| 004 | 04 | 067 | 37 | 152 | 6A | 235 | 9D | 320 | D0 |
| 005 | 05 | 070 | 38 | 153 | 6B | 236 | 9F | 321 | D1 |
| 006 | 06 | 071 | 39 | 154 | 6C | 237 | 9F | 322 | D2 |
| 007 | 07 | 072 | 3A | 155 | 6D | 240 | A0 | 323 | D3 |
| 010 | 08 | 073 | 3B | 156 | 6E | 241 | A1 | 324 | D4 |
| 011 | 09 | 074 | 3C | 157 | 6F | 242 | A2 | 325 | D5 |
| 012 | 0A | 075 | 3D | 160 | 70 | 243 | A3 | 326 | D6 |
| 013 | 0B | 076 | 3E | 161 | 71 | 244 | A4 | 327 | D7 |
| 014 | 0C | 077 | 3F | 162 | 72 | 245 | A5 | 330 | D8 |
| 015 | 0D | 100 | 40 | 163 | 73 | 246 | A6 | 331 | D9 |
| 016 | 0E | 101 | 41 | 164 | 74 | 247 | A7 | 332 | DA |
| 017 | 0F | 102 | 42 | 165 | 75 | 250 | A8 | 333 | DB |
| 020 | 10 | 103 | 43 | 166 | 76 | 251 | A9 | 334 | DC |
| 021 | 11 | 104 | 44 | 167 | 77 | 252 | AA | 335 | DD |
| 022 | 12 | 105 | 45 | 170 | 78 | 253 | AB | 336 | DE |
| 023 | 13 | 106 | 46 | 171 | 79 | 254 | AC | 337 | DF |
| 024 | 14 | 107 | 47 | 172 | 7A | 255 | AD | 340 | E0 |
| 025 | 15 | 110 | 48 | 173 | 7B | 256 | AE | 341 | E1 |
| 026 | 16 | 111 | 49 | 174 | 7C | 257 | AF | 342 | E2 |
| 027 | 17 | 112 | 4A | 175 | 7D | 260 | B0 | 343 | E3 |
| 030 | 18 | 113 | 4B | 176 | 7E | 261 | B1 | 344 | E4 |
| 031 | 19 | 114 | 4C | 177 | 7F | 262 | B2 | 345 | E5 |
| 032 | 1A | 115 | 4D | 200 | 80 | 263 | B3 | 346 | E6 |
| 033 | 1B | 116 | 4E | 201 | 81 | 264 | B4 | 347 | E7 |
| 034 | 1C | 117 | 4F | 202 | 82 | 265 | B5 | 350 | E8 |
| 035 | 1D | 120 | 50 | 203 | 83 | 266 | B6 | 351 | E9 |
| 036 | 1E | 121 | 51 | 204 | 84 | 267 | B7 | 352 | EA |
| 037 | 1F | 122 | 52 | 205 | 85 | 270 | B8 | 353 | EB |
| 040 | 20 | 123 | 53 | 206 | 86 | 271 | B9 | 354 | EC |
| 041 | 21 | 124 | 54 | 207 | 87 | 272 | BA | 355 | ED |
| 042 | 22 | 125 | 55 | 210 | 88 | 273 | BB | 356 | EE |
| 043 | 23 | 126 | 56 | 211 | 89 | 274 | BC | 357 | EF |
| 044 | 24 | 127 | 57 | 212 | 8A | 275 | BD | 360 | F0 |
| 045 | 25 | 130 | 58 | 213 | 8B | 276 | BE | 361 | F1 |
| 046 | 26 | 131 | 59 | 214 | 8C | 277 | BF | 362 | F2 |
| 047 | 27 | 132 | 5A | 215 | 8D | 300 | C0 | 363 | F3 |
| 050 | 28 | 133 | 5B | 216 | 8E | 301 | C1 | 364 | F4 |
| 051 | 29 | 134 | 5C | 217 | 8F | 302 | C2 | 365 | F5 |
| 052 | 2A | 135 | 5D | 220 | 90 | 303 | C3 | 366 | F6 |
| 053 | 2B | 136 | 5E | 221 | 91 | 304 | C4 | 367 | F7 |
| 054 | 2C | 137 | 5F | 222 | 92 | 305 | C5 | 370 | F8 |
| 055 | 2D | 140 | 60 | 223 | 93 | 306 | C6 | 371 | F9 |
| 056 | 2E | 141 | 61 | 224 | 94 | 307 | C7 | 372 | FA |
| 057 | 2F | 142 | 62 | 225 | 95 | 310 | C8 | 373 | FB |
| 060 | 30 | 143 | 63 | 226 | 96 | 311 | C9 | 374 | FC |
| 061 | 31 | 144 | 64 | 227 | 97 | 312 | CA | 375 | FD |
| 062 | 32 | 145 | 65 | 230 | 98 | 313 | CB | 376 | FE |
| | | | | | | | | 377 | FF |

# APPENDICE 5

# DESCRIZIONE DEL MICROCOMPUTER MMD-1

## INTRODUZIONE

In questa appendice, imparerete come viene usato un microprocessore 8080 (o 8080A) per configurare un piccolo microcomputer basato sull'8080. Prenderete in esame i segnali che entrano e lasciano il chip 8080, come si usano i chip ausiliari come l'8224 per controllare le operazioni del microcomputer, e lo sviluppo degli indirizzi, dei dati, e dei bus di controllo, che sono vitali nelle applicazioni relative agli interfacciamenti. Consultate il Manuale Operativo *"Mini Micro-Designer"* MMD-1 della E. & L. Instruments, Inc. Per ulteriori dettagli sull'assemblaggio e le operazioni del microcomputer. (In Italia: Microlem - S.p.A. Milano)

## OBIETTIVI

Alla fine di questa appendice, sarete in grado di:

- Identificare il bus degli indirizzi di memoria, il bus di dati, gli ingressi di controllo, le uscite di controllo e gli ingressi di alimentazione sul microprocessore 8080A a 40 pin.
- Descrivere la funzione di ogni pin sul microprocessore 8080A.
- Descrivere abbastanza dettagliatamente le varie sezioni che compongono il microcomputer MMD-1 basato sull'8080A.

## IL MICROPROCESSORE 8080

Il microprocessore 8080 è un circuito integrato LSI a 40 pin che contiene sedici linee d'indirizzo, otto linee di dati, dieci linee di controllo, quattro collegamenti per l'alimentazione e una coppia di ingressi di clock. La configurazione dei pin e lo schema a blocchi del chip sono mostrati nelle Figure A5-1 e A5-2. Se non avete familiarità con la lettura del numero dei pin sui circuiti integrati, consultate il Capitolo N. 10.

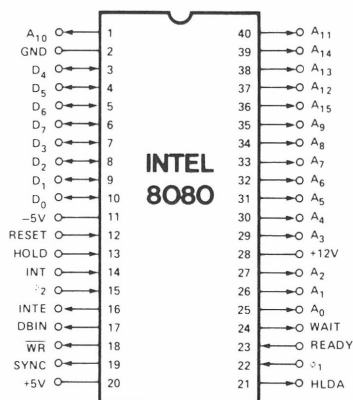



*Figura A5-1.* Configurazione dei pin del microprocessore 8080 a 40 pin.

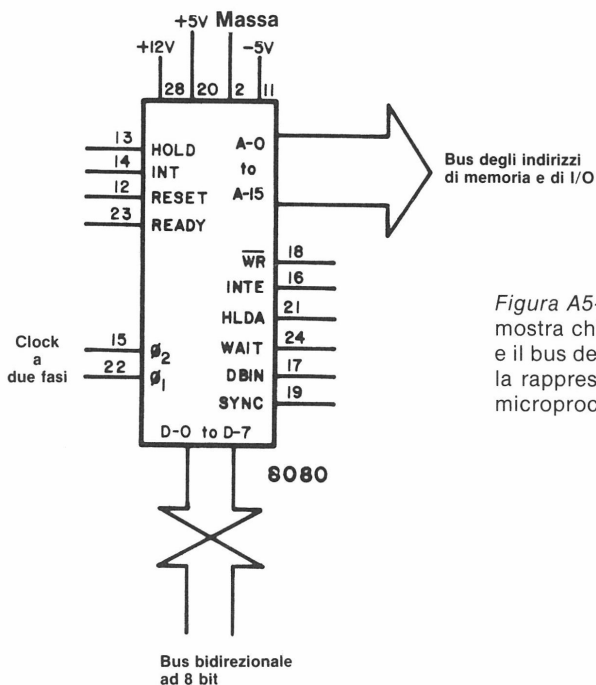



*Figura A5-2.* Schema a blocchi dell'8080, che mostra chiaramente il bus d'indirizzo a 16 bit e il bus dei dati bidirezionale a 8 bit. Questa è la rappresentazione più utile del microprocessore 8080.

# Silicon Gate MOS 8080A

## SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSOR

*The 8080A is functionally and electrically compatible with the Intel® 8080.*

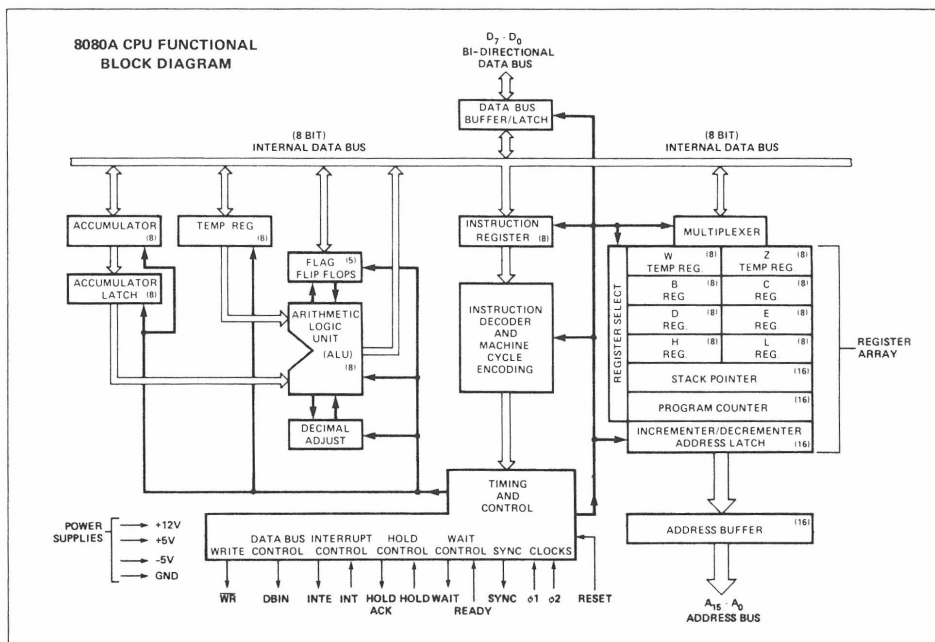
- **TTL Drive Capability**
- **2 μs Instruction Cycle**
- **Powerful Problem Solving Instruction Set**
- **Six General Purpose Registers and an Accumulator**
- **Sixteen Bit Program Counter for Directly Addressing up to 64K Bytes of Memory**
- **Sixteen Bit Stack Pointer and Stack Manipulation Instructions for Rapid Switching of the Program Environment**
- **Decimal,Binary and Double Precision Arithmetic**
- **Ability to Provide Priority Vectored Interrupts**
- **512 Directly Addressed I/O Ports**

The Intel® 8080A is a complete 8-bit parallel central processing unit (CPU). It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process. This offers the user a high performance solution to control and processing applications.

The 8080A contains six 8-bit general purpose working registers and an accumulator. The six general purpose registers may be addressed individually or in pairs providing both single and double precision operators. Arithmetic and logical instructions set or reset four testable flags. A fifth flag provides decimal arithmetic operation.

The 8080A has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter and all of the six general purpose registers. The sixteen bit stack pointer controls the addressing of this external stack. This stack gives the 8080A the ability to easily handle multiple level priority interrupts by rapidly storing and restoring processor status. It also provides almost unlimited subroutine nesting.

This microprocessor has been designed to simplify systems design. Separate 16-line address and 8-line bi-directional data busses are used to facilitate easy interface to memory and I/O. Signals to control the interface to memory and I/O are provided directly by the 8080A. Ultimate control of the address and data busses resides with the HOLD signal. It provides the ability to suspend processor operation and force the address and data busses into a high impedance state. This permits OR-tying these busses with other controlling devices for (DMA) direct memory access or multi-processor operation.

**8080A CPU FUNCTIONAL BLOCK DIAGRAM**

È utile suddividere le funzioni dei pin nelle seguenti categorie: alimentazione, bus degli indirizzi di memoria, bus dei dati bidirezionale, segnali di controllo, e ingressi di clock.

## ALIMENTAZIONE

| | |
|---|---|
| pin 28 | +12 V |
| pin 20 | + 5 V |
| pin 2 | Massa (Ground) |
| pin 11 | − 5 V |

Le tolleranze di tensione sono ±5% rispetto al potenziale di massa. Qualunque alimentazione che fornisce tensioni di ±15 e +5 V e sufficiente corrente può essere adattata al chip 8080 con l'aiuto di regolatori di tensione adeguati.

## INGRESSI DI CLOCK

Il chip 8080 richiede un *clock a due fasi.* Ricordate che un *clock* è un qualunque dispositivo che genera almeno un impulso di clock, o un dispositivo di timing in un sistema che fornisce una continua serie di impulsi di timing. Un clock a due fasi è un dispositivo di timing a due uscite che fornisce due serie continue di impulsi di timing che sono sincronizzati insieme, con un solo impulso di clock della seconda serie che segue sempre un solo impulso di clock della prima serie. L'uso di diagrammi per il calcolo dei tempi, mostrato in Figura A5-3, è utile per spiegare come opera un clock a due fasi:

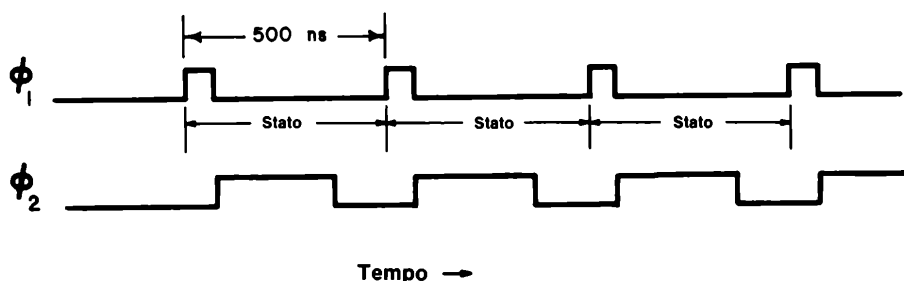



*Figura A5-3.* Grafico degli ingressi di clock a due fasi dell'8080.

Notate che il fronte iniziale della serie $Ø_2$ di impulsi di clock si sovrappongono quasi al fronte finale della serie $Ø_1$ di impulsi. Nelle specifiche dell'8080, è stabilito che la minima larghezza d'impulso per l'impulso di clock $Ø_1$ è 60 ns, mentre per l'impulso di clock $Ø_2$è 220 ns. Le posizioni dei pin sul chip 8080 per i due segnali di clock d'ingresso sono:

| | |
|---|---|
| pin 22 | Fase di clock $Ø_1$ |
| pin 15 | Fase di clock $Ø_2$ |

Chiamiamo questo tipo di dispositivo di clock: *clock a due fasi senza sovrapposizione dei tempi.* Questo non è un clock di livello TTL. Piuttosto, oscilla da 0 V a +12 V. Un clock di questo tipo può essere facilmente generato con un chip clock-generator 8224, messo a disposizione dalla Intel Corporation e da altri costruttori.

**BUS DEGLI INDIRIZZI DI MEMORIA**

Il microprocessore 8080 può indirizzare direttamente fino a 65.536 parole a otto bit di memoria, con l'uso di sedici linee d'indirizzo con uscita three-state, chiamate *"bus d'indirizzi"*. Le posizioni dei pin per il bus possono essere riassunte nel modo seguente:

| | |
|---|---|
| pin 25 | Bit d'indirizzo A0, il bit meno significativo |
| pin 26 | Bit d'indirizzo A1 |
| pin 27 | Bit d'indirizzo A2 |
| pin 29 | Bit d'indirizzo A3 |
| pin 30 | Bit d'indirizzo A4 |
| pin 31 | Bit d'indirizzo A5 |
| pin 32 | Bit d'indirizzo A6 |
| pin 33 | Bit d'indirizzo A7, l'MSB per il codice dispositivo a 8 bit |
| pin 34 | Bit d'indirizzo A8, l'LSB per il codice dispositivo a 8 bit |
| pin 35 | Bit d'indirizzo A9 |
| pin 1 | Bit d'indirizzo A10 |
| pin 40 | Bit d'indirizzo A11 |
| pin 37 | Bit d'indirizzo A12 |
| pin 38 | Bit d'indirizzo A13 |
| pin 39 | Bit d'indirizzo A14 |
| pin 36 | Bit d'indirizzo A15, il bit più significativo |

Si possono usare o i bit da A0 ad A7 o i bit d'indirizzo da A8 ad A15 per fornire il codice dispositivo di I/O per al massimo 25 diversi dispositivi d'ingresso o 256 diversi dispositivi di uscita. Le linee d'indirizzo si innestano direttamente in circuiti di decodifica che usano il 74L42, 74L154 o altri chip di decodifica.

**BUS DI DATI BIDIREZIONALE**

Il microprocessore 8080 è un microprocessore a otto bit, il che significa che esistono un accumulatore a otto bit, molti registri aggiuntivi a otto bit e un bus di dati bidirezionale a otto bit. Dato che il bus è bidirezionale, i dati possono essere o inseriti o messi in uscita sulle stesse otto linee. Il bus di dati è il bus di comunicazione principale fra la CPU nel microprocessore e la memoria o i dispositivi di I/O.

Questo è un bus bidirezionale three-state in cui le posizioni dei pin sono le seguenti:

| | |
|---|---|
| pin 10 | Bit del bus di dati D0, il bit meno significativo |
| pin 9 | Bit del bus di dati D1 |
| pin 8 | Bit del bus di dati D2 |
| pin 7 | Bit del bus di dati D3 |
| pin 3 | Bit del bus di dati D4 |
| pin 4 | Bit del bus di dati D5 |
| pin 5 | Bit del bus di dati D6 |
| pin 6 | Bit del bus di dati D7, il bit più significativo |

**SEGNALI DI CONTROLLO**

I pin dei segnali di controllo determinano come funziona il microprocessore in un sistema microcomputer. Discutendo le funzioni di questi pin, abbiamo preferito non utilizzare termini come *T1, T2, T3,* $T_w$, *ciclo di fetch,* $M_1$ ed altri. Forniamo le identificazioni dei pin e le descrizioni nell'elenco che segue per farvi riferimento in futuro, quando capirete meglio l'operatività del microcomputer 8080.

Notate che non vedrete né il segnale di controllo $\overline{IN}$ né $\overline{OUT}$ nell'elenco che segue. La ragione stà nel fatto che queste due funzioni sono generate come bit di stato, che sono memorizzate esternamente usando chip di latch quali il 74174 o 8212, e vengono usate per generare gli impulsi di sincronizzazione $\overline{IN}$ e $\overline{OUT}$ precedentemente nominati. Lo stesso discorso è valido per la generazione dei segnali di controllo, $\overline{MEMR}$, $\overline{MEMW}$, e $\overline{INTA}$.

I quattro pin d'*ingresso* di controllo sul microprocessore 8080 sono:

| | |
|---|---|
| pin 12 (ingresso) | RESET. Un livello logico 0 a questo ingresso azzererà il registro contatore di programma e permetterà al programma di iniziare alla locazione di memoria HI = 000 e LO = 000. I flag INTE e HLDA vengono anch'essi resettati, ma i flag di condizione, il registro accumulatore, il registro stack pointer e i registri universali non vengono azzerrati. |
| pin 14 (ingresso) | INT, o INTERRUPT REQUEST. Un livello logico 1 a questo ingresso genererà una richiesta di interruzione che la CPU riconosce alla fine dell'istruzione in corso, o all'arresto. Se la CPU non è nello stato HOLD, o se il flip-flop di abilitazione all'interruzione è resettato, (cioè a livello logico 0), la richiesta di interruzione non verrà accettata. |
| pin 23 (ingresso) | READY. Un livello logico 1 indica all'8080 che sulle linee del bus di dati sono disponibili una memoria valida o i dati in ingresso, da D0 a D7. Questo segnale si usa per sincronizzare la CPU con memoria più lenta o con dispositivi di I/O. |

Se, dopo aver mandato un indirizzo sul bus d'indirizzo, l'8080 non riceve un ingresso READY di livello logico 1, il microprocessore entrerà in uno stato di WAIT, in cui resta fino a che la linea READY è a livello logico 0. Questo ingresso può essere anche usato per far avanzare la CPU passo a passo.

pin 13 (ingresso) HOLD. Questo pin d'ingresso richiede alla CPU di entrare in stato di HOLD, il che permette ad un dispositivo esterno di ottenere il controllo dei bus dei dati e degli indirizzi non appena l'8080 ha completato l'uso di questi bus per il ciclo macchina in corso. Una volta che la CPU entra in stato di HOLD, il bus degli indirizzi e il bus di dati bidirezionali sono nel loro stato di alta impedenza. HOLD è attivo nello stato logico 1.

La CPU riconosce lo stato HOLD con HLDA, o pin di uscita HOLD ACKNOWLEDGE. HOLD è riconosciuto a due condizioni: (1) la CPU è in stato di HALT o (2) la CPU è nello stato T2 o $T_w$ e il segnale READY è a livello logico 1.

Questo per gli ingressi di controllo. Ora riassumiamo le uscite di controllo, molte delle quali sono flag, che si possono definire come:

*Flag* In un computer, un'indicazione che una particolare operazione è stata terminata. Un flag è di solito un flip-flop che si può o settare o azzerare in risposta alle operazioni che avvengono nel computer.

I sei pin di uscita di controllo sul microprocessore 8080 sono:

pin 24 (uscita) WAIT. Il segnale di uscita wait riconosce che la CPU è in stato di WAIT. Quando è in stato di WAIT, questo pin è a livello logico 1.

pin 18 (uscita) $\overline{WR}$, o WRITE. Questo pin di uscita si usa per il controllo dell'uscita di scrittura in memoria o di I/O. Quando questo pin è a livello logico 0, i dati sul bus di dati sono stabili e possono essere scritti in memoria o in un dispositivo di I/O.

pin 21 (uscita) HLDA, o HOLD ACKNOWLEDGE. Questo pin va a livello logico 1 in risposta al segnale d'ingresso HOLD. Esso indica che i dati e i bus degli indirizzi andranno ad uno stato di alta impedenza. Il segnale HLDA inizia in uno di questi due momenti: (1) a T3 per lettura in memoria o ingresso, o (2) Il periodo di clock che segue T3 per la scrittura in memoria o le operazioni di uscita.

pin 16 (uscita) INTE, o INTERRUPT ENABLE. Questo pin indica i contenuti del flip-flop interno di abilitazione all'interruzione. T è il flip-flop che può essere settato o azzerato con le istruzioni di abilitazione e disabilitazione all'interruzione (EI e DI, rispettivamente) e impedisce all'interruzione di venire accettata dalla CPU quando il flip-flop è azzerato. Il flip-flop è azzerato automaticamente (disabilitando così ulteriori interruzioni) nel momento T1 del ciclo di fetch dell'istruzione (M1) quando viene accettata un'interruzione. Il flip-flop è azzerato anche dall'ingresso di controllo RESET.

pin 19 (uscita) — SYNC, o SYNCRONIZING SIGNAL. Il pin SYNC fornisce un segnale di livello logico 1 per indicare l'inizio di ogni ciclo macchina.

pin 17 (uscita) — DBIN, o DATA BUS IN. Quando questo pin va a livello logico 1, indica ai circuiti esterni che il bus dei dati è nel modo di operazione d'ingresso. Questo pin dovrebbe essere usato per abilitare il gate dei dati sul bus dei dati dell'8080, dalla memoria o dai dispositivi di I/O.

## CLOCK GENERATOR/DRIVER 8224

Nei primi sistemi di microcomputer basati sull'8080, gli ingressi di temporizzazione erano forniti da driver a transistor, da clock driver MOS o da buffer TTL a collettore aperto. Tutti lavoravano abbastanza bene, ma complicavano la progettazione. Un recente chip di interfaccia per l'8080, il clock generator/driver 8224, contiene un oscillatore ed un clock generator/driver. Tutto quello che dovete aggiungere è un cristallo appropriato nonché le tensioni di alimentazione di +5 e +12 V. Dato che l'8224 dividerà la frequenza del cristallo per nove, avrete bisogno di un cristallo a 6,750 MHz per produrre un'uscita di temporizzazione di 750 kHz dal clock generator.

Le specifiche della Intel per il clock generator/driver 8224 sono mostrate nelle pagine seguenti. La descrizione funzionale del chip è ottima, perciò non occorre che la ripetiamo. Osservate come il circuito contatore divisore per nove all'interno del chip viene usato per generare le fasi di temporizzazione individuali $\emptyset_1$ o $\emptyset_2$ che "oscillano" fra +12 V e massa.

Gli ingressi e le uscite dal chip 8224 possono essere riassunti nel modo seguente:

pin 14 e 15 — XTAL1 e XTAL2. Il cristallo è collegato a questi due pin.

pin 13 — TANK. Usato per i cristalli che hanno un guadagno molto più basso di quelli operanti sulla frequenza fondamentale.

pin 2 (ingresso) — $\overline{\text{RESIN}}$. Con l'aiuto di un circuito di trigger di Schmitt, che è interno al chip, ed una rete RC esterna, questo ingresso converte una transizione lenta nell'alimentazione in un fronte veloce e pulito che resetta il microprocessore 8080. Un impulso di uscita RESET può essere ottenuto anche applicando a $\overline{\text{RESIN}}$ un livello logico 0.

pin 1 (uscita) — RESET. Un livello logico 0 da questo pin di uscita applicato all'ingresso RESET del chip 8080 resetterà l'8080.

pin 3 (ingresso) — RDYN. Accetta una richiesta di attesa asincrona e la sincronizza per produrre un impulso di uscita READY che viene inserito nel chip 8080.

pin 4 (uscita) — READY. Un livello logico 1 a questo pin di uscita indica all'8080 che sul bus di dati sono disponibili una memoria valida o i dati di ingresso.

# Schottky Bipolar 8224

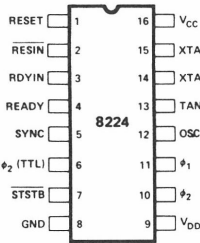
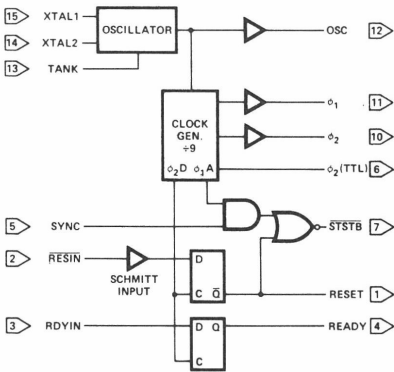
## CLOCK GENERATOR AND DRIVER FOR 8080A CPU

- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**

The 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer, to meet a variety of system speed requirements.

Also included are circuits to provide power-up reset, advance status strobe and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

### PIN CONFIGURATION

### BLOCK DIAGRAM

### PIN NAMES

| | |
|---|---|
| RESIN | RESET INPUT |
| RESET | RESET OUTPUT |
| RDYIN | READY INPUT |
| READY | READY OUTPUT |
| SYNC | SYNC INPUT |
| STSTB | STATUS STB (ACTIVE LOW) |
| $\phi_1$ | 8080 |
| $\phi_2$ | CLOCKS |

| | |
|---|---|
| XTAL 1 | CONNECTIONS |
| XTAL 2 | FOR CRYSTAL |
| TANK | USED WITH OVERTONE XTAL |
| OSC | OSCILLATOR OUTPUT |
| $\phi_2$ (TTL) | $\phi_2$ CLK (TTL LEVEL) |
| $V_{CC}$ | +5V |
| $V_{DD}$ | +12V |
| GND | 0V |

## FUNCTIONAL DESCRIPTION

### General

The 8224 is a single chip Clock Generator/Driver for the 8080A CPU. It contains a crystal-controlled oscillator, a "divide by nine" counter, two high-level drivers and several auxiliary logic functions.

### Oscillator

The oscillator circuit derives its basic operating frequency from an external, series resonant, fundamental mode crystal. Two inputs are provided for the crystal connections (XTAL1, XTAL2).

The selection of the external crystal frequency depends mainly on the speed at which the 8080A is to be run at. Basically, the oscillator operates at 9 times the desired processor speed.

A simple formula to guide the crystal selection is:

$$\text{Crystal Frequency} = \frac{1}{t_{CY}} \text{ times } 9$$

Example 1: (500ns $t_{CY}$)
2mHz times 9 = 18mHz*

Example 2: (800ns $t_{CY}$)
1.25mHz times 9 = 11.25mHz

Another input to the oscillator is TANK. This input allows the use overtone mode crystals. This type of crystal generally has much lower "gain" than the fundamental type so an external LC network is necessary to provide the additional "gain" for proper oscillator operation. The external LC network is connected to the TANK input and is AC coupled to ground. See Figure 4.

The formula for the LC network is:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

The output of the oscillator is buffered and brought out on OSC (pin 12) so that other system timing signals can be derived from this stable, crystal-controlled source.

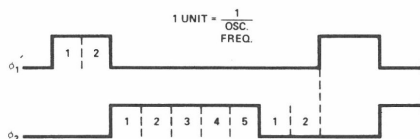
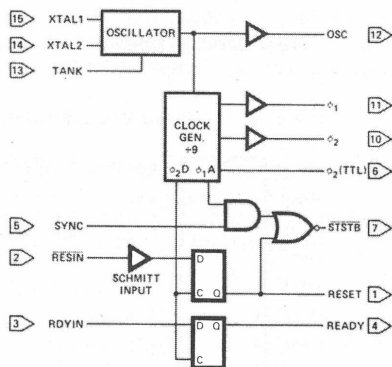
*When using crystals above 10mHz a small amount of frequency "trimming" may be necessary. The addition of a small capacitance (3pF - 10pF) in series with the crystal will accomplish this function.

### Clock Generator

The Clock Generator consists of a synchronous "divide by nine" counter and the associated decode gating to create the waveforms of the two 8080A clocks and auxiliary timing signals.

The waveforms generated by the decode gating follow a simple 2-5-2 digital pattern. See Figure 2. The clocks generated; phase 1 and phase 2, can best be thought of as consisting of "units" based on the oscillator frequency. Assume that one "unit" equals the period of the oscillator frequency. By multiplying the number of "units" that are contained in a pulse width or delay, times the period of the oscillator frequency, the approximate time in nanoseconds can be derived.

The outputs of the clock generator are connected to two high level drivers for direct interface to the 8080A CPU. A TTL level phase 2 is also brought out $\phi_2$ (TTL) for external timing purposes. It is especially useful in DMA dependant activities. This signal is used to gate the requesting device onto the bus once the 8080A CPU issues the Hold Acknowledgement (HLDA).

Several other signals are also generated internally so that optimum timing of the auxiliary flip-flops and status strobe ($\overline{\text{STSTB}}$) is achieved.

# SCHOTTKY BIPOLAR 8224

## $\overline{\text{STSTB}}$ (Status Strobe)

At the beginning of each machine cycle the 8080A CPU issues status information on its data bus. This information tells what type of action will take place during that machine cycle. By bringing in the SYNC signal from the CPU, and gating it with an internal timing signal ($\phi$1A), an active low strobe can be derived that occurs at the start of each machine cycle at the earliest possible moment that status data is stable on the bus. The $\overline{\text{STSTB}}$ signal connects directly to the 8228 System Controller.

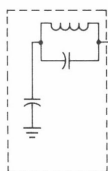
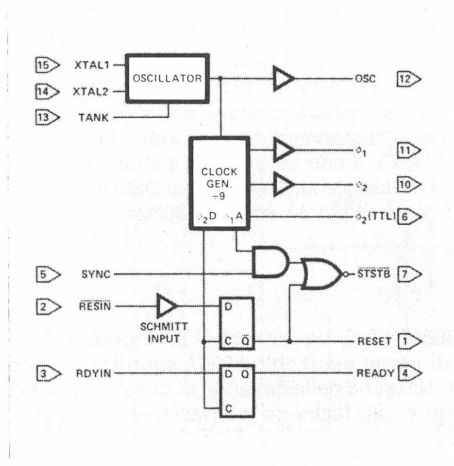
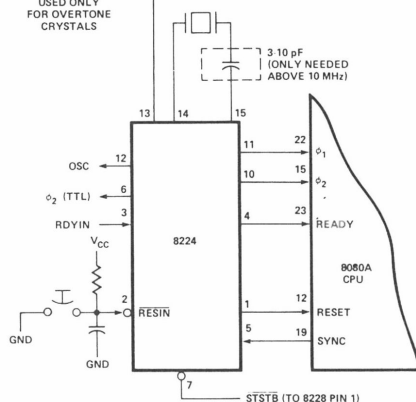
The power-on Reset also generates $\overline{\text{STSTB}}$, but of course, for a longer period of time. This feature allows the 8228 to be automatically reset without additional pins devoted for this function.

## Power-On Reset and Ready Flip-Flops

A common function in 8080A Microcomputer systems is the generation of an automatic system reset and start-up upon initial power-on. The 8224 has a built in feature to accomplish this feature.

An external RC network is connected to the $\overline{\text{RESIN}}$ input. The slow transition of the power supply rise is sensed by an internal Schmitt Trigger. This circuit converts the slow transition into a clean, fast edge when its input level reaches a predetermined value. The output of the Schmitt Trigger is connected to a "D" type flip-flop that is clocked with $\phi$2D (an internal timing signal). The flip-flop is synchronously reset and an active high level that complies with the 8080A input spec is generated. For manual switch type system Reset circuits, an active low switch closing can be connected to the $\overline{\text{RESIN}}$ input in addition to the power-on RC net-network.

The READY input to the 8080A CPU has certain timing specifications such as "set-up and hold" thus, an external synchronizing flip-flop is required. The 8224 has this feature built-in. The RDYIN input presents the asynchronous "wait request" to the "D" type flip-flop. By clocking the flip-flop with $\phi$2D, a synchronized READY signal at the correct input level, can be connected directly to the 8080A.

The reason for requiring an external flip-flop to synchronize the "wait request" rather than internally in the 8080 CPU is that due to the relatively long delays of MOS logic such an implementation would "rob" the designer of about 200ns during the time his logic is determining if a "wait" is necessary. An external bipolar circuit built into the clock generator eliminates most of this delay and has no effect on component count.

| | |
|---|---|
| pin 5 (ingresso) | SYNC. Il pin SYNC sul chip 8080 fornisce un ingresso di sincronizzazione al chip 8224 per indicare l'inizio di ogni ciclo macchina. |
| pin 11 (uscita)<br>pin 10 (uscita) | $\emptyset_1$ e $\emptyset_2$. Il clock a due fasi utilizzato dall'8080.<br>Queste due uscite "oscillano" fra +12 V e potenziale di massa; non sono normali uscite TTL. |
| pin 6 (uscita) | $\emptyset_2$ (TTL). Questo è un clock di uscita TTL che ha la stessa frequenza e le stesse caratteristiche di timing, di $\emptyset_2$. È usato per temporizzazioni esterne. |
| pin 7 (uscita) | $\overline{STSTB}$. Status Strobe Output. Questo pin di uscita si usa per eseguire un latch sui bit di stato che appaiono sul bus di dati bidirezionale. |
| pin 12 (uscita) | OSC. Segnale di uscita dell'oscillatore a quarzo; è sottoposto a buffer e si può usare per generare altri segnali di temporizzazione. |

Dovrebbe esere chiaro che il clock/generator driver 8224 è ben progettato per le sue particolari funzioni. I collegamenti fra questo e il microprocessore 8080 sono diretti e non richiedono invertitori intermedi, gate, o flip-flop. Non c'è molto incentivo ad usare i circuiti di driver a transistor, i clock driver MOS o i buffer TTL a collettore aperto. L'alimentazione per il chip 8224 è già disponibile, dato che per l'8080 sono necessari sia i +5 V che i +12 V.

## IL MICROCOMPUTER MMD-1

Nella Figura A5-4 è mostrata la sezione del processore centrale del microcomputer MMD-1. La figura è gentilmente fornita dalla rivista Radio-Electronics, che ha descritto il microcomputer MMD-1 nei numeri di Maggio, Giugno e Luglio 1976. Vorremmo ora esaminare i chip che compongono il circuito nonchè il flusso di segnali fra di loro. Il nostro obiettivo è di mostrare che un microcomputer è un dispositivo di facile utilizzo e che non dovreste farvi intimidire da uno schema come quello dato in Figura A5-4.

### ALIMENTAZIONE

Si suppone che siano disponibili le alimentazioni di +5 V, −12 V e +12 V richieste. Esse sono relativamente non costose, ma non fidatevi delle alimentazioni molto economiche. Le tensioni intermedie di −5 V e −9 V richieste dal nostro microcomputer si ricavano facilmente da regolatori di tensione come quelli della serie LM320, o con diodi zener a shunt, come si vede nell'angolo in alto a sinistra della Figura A5-4. I diodi zener 1N746 (chip D26) e 1N751A (chip D25) forniscono −9 V per le EPROM e −5 V per il pin 11 sul chip 8080A.

### MICROPROCESSORE 8080A

I pin di uscita individuali del microprocessore 8080 hanno un fan-out di un ingresso TTL low-power, circa 0,16 mA. Le specifiche dei pin di uscita per il chip 8080A sono 1,9 mA per ogni pin di uscita, fan-out di poco maggiore di 1. Nessuna delle capacità di questi fan-out è buona, ma chiaramente l'8080A è un chip superiore, più facile da interfacciare.

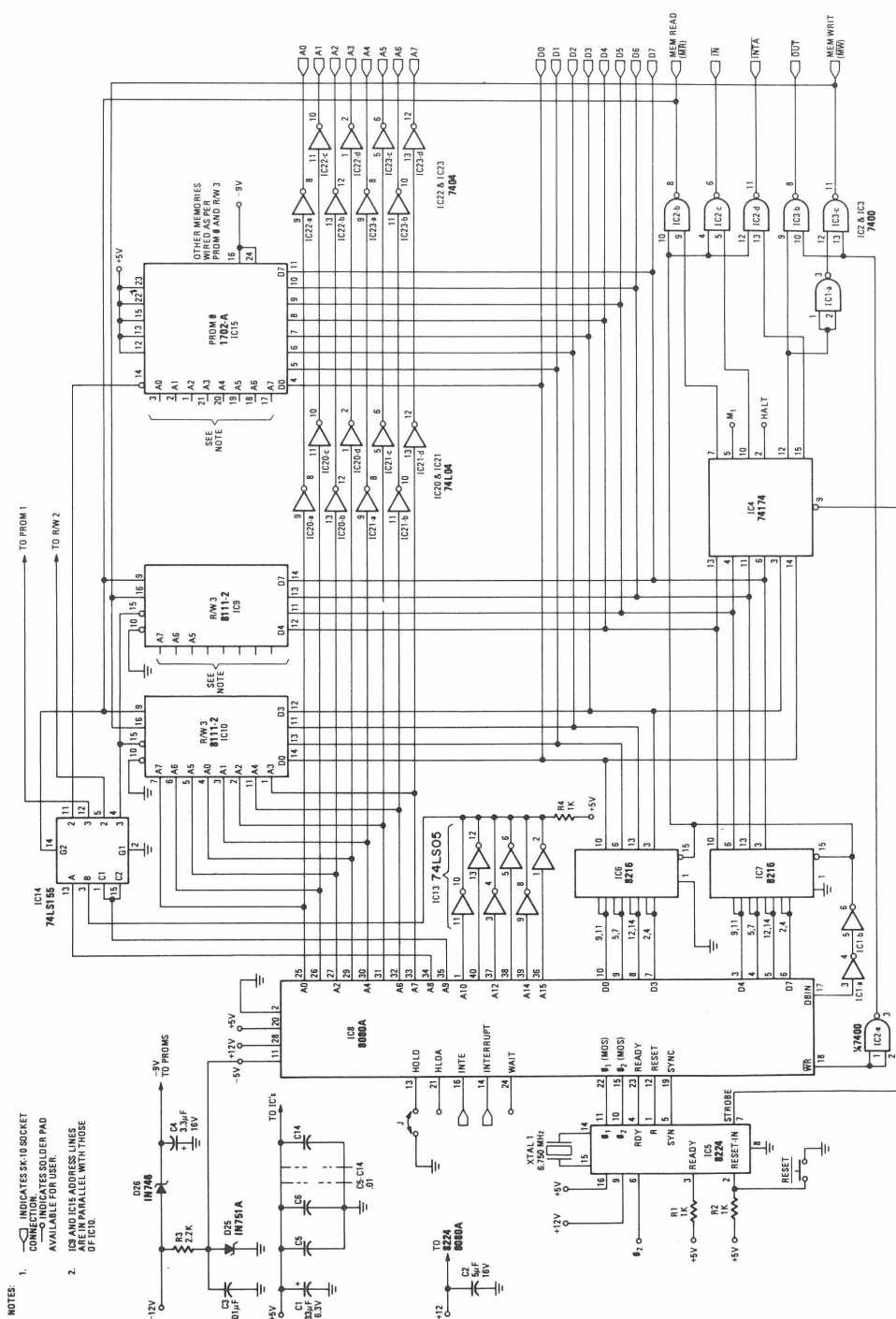



*Figura A5-4.* La CPU, la memoria, e le sezioni di controllo del microcomputer MMD-1.

Per questa ragione, lo usiamo nel microcomputer MMD-1. Anche un fan-out di 1 è insufficiente per guidare i chip di memoria richiesti e i latch di uscita. Di conseguenza, sono necessari anche i bus driver 8216. Di questo parleremo più avanti nella parte riguardante i bus driver. Le specifiche per il chip 8080A sono già state indicate.

## LINEE DI CONTROLLO

La sezione di controllo del microcomputer MMD-1 è mostrata nell'angolo in basso a sinistra della Figura A5-4 e consiste del clock generator/driver 8224 collegato direttamente all'8080A, di una coppia di resistori da 1 kΩ, di uno switch di reset e di un cristallo a 6,750 MHz. Le linee di controllo rimanenti sul chip 8080A, quelle non collegate al chip 8224, sono HOLD, HLDA, INTE, INTERRUPT, WAIT, $\overline{WR}$ e DBIN. Cinque di queste linee non vengono usate nel microcomputer MMD-1, ma sono disponibili se volete sperimentarle. L'ingresso HOLD vi permette di portare il chip 8080 in stato di hold e di disabilitare il bus degli indirizzi e dei dati. L'uscita di controllo HLDA riconosce l'esistenza di uno stato di hold. L'ingresso INTERRUPT vi permette di interrompere l'esecuzione del programma dell'8080A, ammesso che il flip-flop di interrupt all'ingresso del chip 8080A sia abilitato. Se è abilitato, l'uscita INTE è a livello logico 1. Infine, l'uscita WAIT permette al chip 8080A di segnalare che non è pronto o che sta aspettando che avvenga qualcosa all'esterno. Se l'ingresso HOLD al pin 13 dell'8080A non viene usato, deve essere messo a massa. Ciò si esegue facilmente con l'aiuto di un ponticello, come mostra la Figura A5-4.

Le ultime due linee di controllo sono entrambe delle uscite. Il segnale WRITE ($\overline{WR}$) è attivo quando è a livello logico 0 e indica che l'8080A stà inviando i dati a qualche dispositivo. Il segnale rimanente, DATA BUS IN (DBIN), indica che il bus di dati viene usato in quel momento per l'ingresso dei dati. È attivo nello stato logico 1. Nell'8080A e in altri microprocessori analoghi, il bus di dati è bidirezionale, cioè il trasferimento dei dati da e verso il chip avviene sugli stessi collegamenti elettrici. È necessario usare con cura il bus di dati perché il flusso dei dati avvenga in modo corretto. Questa capacità è compresa nel microprocessore stesso, ma dobbiamo accertarci che i nostri dispositivi esterni non tentino di mettere i loro dati sul bus nello stesso momento, o quando sta tendando di farlo qualche altro dispositivo o forse lo stesso 8080A. In ogni istante, solo un dispositivo dovrebbe trasmettere i dati sul bus di dati.

## BUS DRIVER

Allo scopo di pilotare i chip di memoria e i latch di uscita del microcomputer MMD-1, è necessario un fan-out di almeno dieci per ogni linea di uscita sul bus di dati. Inoltre, va mantenuto il carattere bidirezionale del bus di dati. Il dispositivo usato per realizzare tali obiettivi è il bus driver bidirezionale parallelo a 4 bit 8216 della Intel Corporation, le cui specifiche sono mostrate , per concessione della Intel Corporation, su alcune pagine seguenti. Consideriamo l'uscita $DB_0$ nello schema logico dell'8216. Ecco la tabella della verità:

| $\overline{DIEN}$ | $\overline{CS}$ | |
|---|---|---|
| 0 | 0 | $DI_0 \rightarrow DB_0$, cioè i dati sono messi in uscita dal chip 8080A |
| 1 | 0 | $DB_0 \rightarrow DI_0$, cioè i dati sono inseriti nel chip 8080A |
| 0 | 1 | Stato di alta impedenza, cioè il chip è disabilitato |
| 1 | 1 | Stato di alta impedenza, cioè il chip è disabilitato |

# Schottky Bipolar 8216/8226

## 4 BIT PARALLEL BIDIRECTIONAL BUS DRIVER

- **Data Bus Buffer Driver for 8080 CPU**
- **Low Input Load Current — .25 mA Maximum**
- **High Output Drive Capability for Driving System Data Bus**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU**
- **Three State Outputs**
- **Reduces System Package Count**

The 8216/8226 is a 4-bit bi-directional bus driver/receiver.

All inputs are low power TTL compatible. For driving MOS, the DO outputs provide a high 3.65V $V_{OH}$, and for high capacitance terminated bus structures, the DB outputs provide a high 50mA $I_{OL}$ capability.

A non-inverting (8216) and an inverting (8226) are available to meet a wide variety of applications for buffering in microcomputer systems.

### PIN CONFIGURATION

8216/8226

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | $\overline{CS}$ | 16 | $V_{CC}$ |
| 2 | $DO_0$ | 15 | $\overline{DIEN}$ |
| 3 | $DB_0$ | 14 | $DO_3$ |
| 4 | $DI_0$ | 13 | $DB_3$ |
| 5 | $DO_1$ | 12 | $DI_3$ |
| 6 | $DB_1$ | 11 | $DO_2$ |
| 7 | $DI_1$ | 10 | $DB_2$ |
| 8 | GND | 9 | $DI_2$ |

### PIN NAMES

| | |
|---|---|
| $DB_0$-$DB_3$ | DATA BUS BI-DIRECTIONAL |
| $DI_0$-$DI_3$ | DATA INPUT |
| $DO_0$-$DO_3$ | DATA OUTPUT |
| $\overline{DIEN}$ | DATA IN ENABLE DIRECTION CONTROL |
| $\overline{CS}$ | CHIP SELECT |

### LOGIC DIAGRAM 8216

### LOGIC DIAGRAM 8226

# SCHOTTKY BIPOLAR 8216/8226

## FUNCTIONAL DESCRIPTION

Microprocessors like the 8080 are MOS devices and are generally capable of driving a single TTL load. The same is true for MOS memory devices. While this type of drive is sufficient in small systems with few components, quite often it is necessary to buffer the microprocessor and memories when adding components or expanding to a multi-board system.

The 8216/8226 is a four bit bi-directional bus driver specifically designed to buffer microcomputer system components.

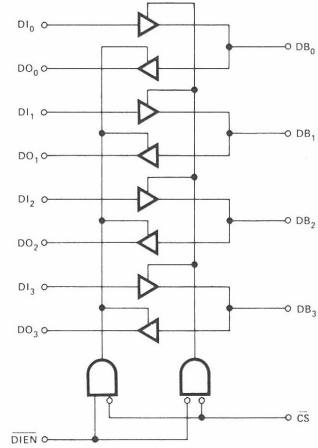
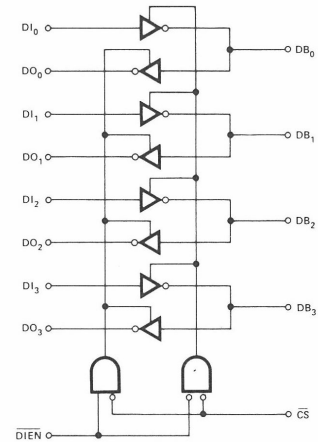
### Bi-Directional Driver

Each buffered line of the four bit driver consists of two separate buffers that are tri-state in nature to achieve direct bus interface and bi-directional capability. On one side of the driver the output of one buffer and the input of another are tied together (DB), this side is used to interface to the system side components such as memories, I/O, etc., because its interface is direct TTL compatible and it has high drive (50mA). On the other side of the driver the inputs and outputs are separated to provide maximum flexibility. Of course, they can be tied together so that the driver can be used to buffer a true bi-directional bus such as the 8080 Data Bus. The DO outputs on this side of the driver have a special high voltage output drive capability (3.65V) so that direct interface to the 8080 and 8008 CPUs is achieved with an adequate amount of noise immunity (350mV worst case).

### Control Gating $\overline{\text{DIEN}}$, $\overline{\text{CS}}$

The $\overline{\text{CS}}$ input is actually a device select. When it is "high" the output drivers are all forced to their high-impedance state. When it is at "zero" the device is selected (enabled) and the direction of the data flow is determined by the $\overline{\text{DIEN}}$ input.

The $\overline{\text{DIEN}}$ input controls the direction of data flow (see Figure 1) for complete truth table. This direction control is accomplished by forcing one of the pair of buffers into its high impedance state and allowing the other to transmit its data. A simple two gate circuit is used for this function.

The 8216/8226 is a device that will reduce component count in microcomputer systems and at the same time enhance noise immunity to assure reliable, high performance operation.

(a) 8216

(b) 8226

| $\overline{\text{DIEN}}$ | $\overline{\text{CS}}$ | |
|---|---|---|
| 0 | 0 | DI → DB |
| 1 | 0 | DB → DO |
| 0 | 1 | HIGH IMPEDANCE |
| 1 | 1 | HIGH IMPEDANCE |

**Figure 1. 8216/8226 Logic Diagrams**

# SCHOTTKY BIPOLAR 8216/8226

## APPLICATIONS OF 8216/8226

### 8080 Data Bus Buffer

The 8080 CPU Data Bus is capable of driving a single TTL load and is more than adequate for small, single board systems. When expanding such a system to more than one board to increase I/O or Memory size, it is necessary to provide a buffer. The 8216/8226 is a device that is exactly fitted to this application.

Shown in Figure 2 are a pair of 8216/8226 connected directly to the 8080 Data Bus and associated control signals. The buffer is bi-directional in nature and serves to isolate the CPU data bus.

On the system side, the DB lines interface with standard semiconductor I/O and Memory components and are completely TTL compatible. The DB lines also provide a high drive capability (50mA) so that an extremely large system can be dirven along with possible bus termination networks.

On the 8080 side the DI and DO lines are tied together and are directly connected to the 8080 Data Bus for bi-directional operation. The DO outputs of the 8216/8226 have a high voltage output capability of 3.65 volts which allows direct connection to the 8080 whose minimum input voltage is 3.3 volts. It also gives a very adequate noise margin of 350mV (worst case).

The control inputs to 8216/8226 ($\overline{\text{CS}}$, $\overline{\text{DIEN}}$) are connected directly to the 8080. $\overline{\text{DIEN}}$ is tied to DBIN so that proper bus flow is maintained, and $\overline{\text{CS}}$ is tied to HLDA so that the system side Data Bus will be 3-stated when a Hold request has been acknowledged during a DMA activity.

### Memory and I/O Interface to a Bi-directional Bus

In large microcomputer systems it is often necessary to provide Memory and I/O with their own buffers and at the same time maintain a direct, common interface to a bi-directional Data Bus. The 8216/8226 has separated data in and data out lines on one side and a common bi-directional set on the other to accomodate such a function.

Shown in Figure 3 is an example of how the 8216/8226 is used in this type of application.

The interface to Memory is simple and direct. The memories used are typically Intel® 8102, 8102A, 8101 or 8107A and have separate data inputs and outputs. The DI and DO lines of the 8216/8226 tie to them directly and under control of the $\overline{\text{MEMR}}$ signal, which is connected to the $\overline{\text{DIEN}}$ input, an interface to the bi-directional Data Bus is maintained.

The interface to I/O is similar to Memory. The I/O devices used are typically Intel® 8255s, and can be used for both input and output ports. The $\overline{\text{I/O R}}$ signal is connected directly to the $\overline{\text{DIEN}}$ input so that proper data flow from the I/O device to the Data Bus is maintained.

The 8216/8226 can be used in a wide variety of other buffering functions in microcomputer systems such as Address Bus Drivers, Drivers to peripheral devices such as printers, and as Drivers for long length cables to other peripherals or systems.

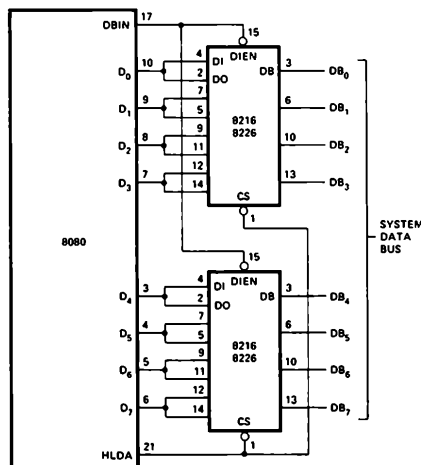



Figure 2. 8080 Data Bus Buffer.

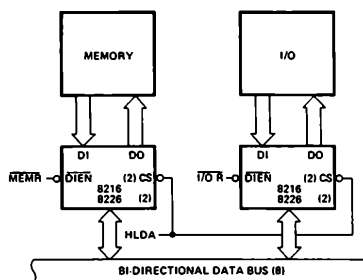



Figure 3. Memory and I/O Interface to a Bi-Directional Bus.

In altre parole, quando $\overline{DIEN}$ è a livello logico 0 e il chip è abilitato, il chip 8216 agisce come buffer d'ingresso. Quando $\overline{DIEN}$ è a livello logico 1 e il chip è abilitato, l'8216 agisce come buffer di uscita, ma *non* come buffer/latch di uscita.

La sezione del driver del bus del microcomputer MMD-1 è mostrata nell'angolo in basso a sinistra della Figura A5-4, alla destra del chip 8080A. Osservate che DBIN è collegato a $\overline{DIEN}$ (pin 15 sul chip 8216) e che ogni chip 8216 è abilitato in modo permanente. La tabella della verità relativa a DBIN e $\overline{DIEN}$ è,

| DBIN | $\overline{DIEN}$ | |
|---|---|---|
| 0 | 0 | I dati sono messi in uscita dal chip 8080A; DBIN = 0, e perciò il bus di dati non si presenta nel modo di funzionamento di ingresso. |
| 1 | 1 | I dati sono inseriti nel chip 8080A; DBIN = 1, perciò il bus di dati si presenta nel modo di funzionamento di ingresso. |

Secondo le specifiche della Intel Corporation per l'8216, la corrente di uscita massima relativa ad un livello logico 0, è 125 mA, che è una capacità di pilotaggio sostanzialmente valida.

**Informazioni di stato**

Se studiate attentamente le specifiche della Intel Corporation per il microprocessore 8080A, osserverete che certi importanti segnali di controllo non sono presenti sul chip stesso. Compresi fra questi sono la lettura in memoria ($\overline{MEMR}$), la scrittura in memoria ($\overline{MEMW}$), l'ingresso ($\overline{IN}$), l'uscita ($\overline{OUT}$), e riconoscimento d'interruzione ($\overline{INTA}$). Per generare tali segnali di controllo, il chip 8080A usa una tecnica "d'avanguardia": dato che il bus di dati non viene usato sempre per il trasferimento dei dati, l'8080A può usare il bus per trasferire le informazioni di controllo aggiuntive. *Tali informazioni vengono generate molto presto nel ciclo macchina per permettere al microcompter di usare tali segnali di controllo per facilitare il trasferimento dei dati da e verso la memoria e i dispositivi di I/O.*

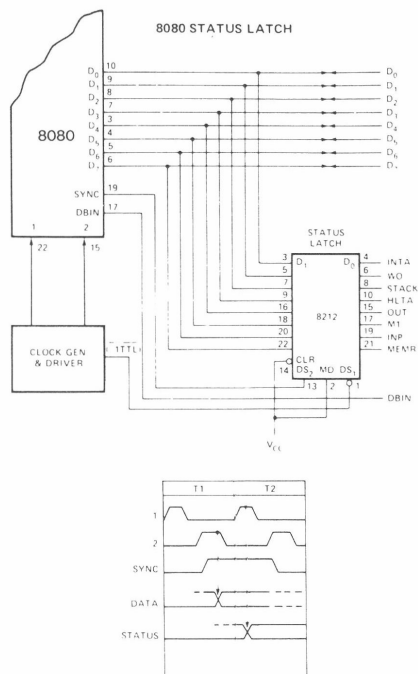
L'informazione di stato appare sul bus di dati per un lasso di tempo molto breve, circa 1,33 $\mu$s, per un sistema 8080 che opera ad una frequenza di 750 kHz. Dato che l'informazione va usata in un secondo tempo, deve essere eseguito un latch. Il SYSTEM STROBE ($\overline{STSTB}$) è generato al pin 7 sul chip 8224 nel momento giusto per eseguire il latch, o catturare, l'informazione di stato. Notate che il segnale $\overline{STSTB}$ è generato dal segnale di temporizzazione $\emptyset_1$ del sistema e il segnale SYNC dall'8080A. Si può usare qualunque tipo di latch a 8 bit. Nella Figura A5-4, sotto la metà, viene usato un latch positive edge triggered a 6 bit; questo è sottoposto a clock al pin 9.

Tutti gli otto bit sul bus di dati forniscono vari tipi di informazioni di stato, ma non sono necessari tutti e otto. Le informazioni fornite dalla Intel Corporation sui bit di stato e i tipi di cicli macchina sono date nella pagina seguente. Nel microcomputer MMD-1, i segnali di stato $\overline{WO}$ e STACK sono ignorati dato che non sono molto utili. Per HLTA e M1 viene eseguito un latch, ma anch'essi non sono usati. I segnali di stato importanti sono INTA (riconoscimento d'interruzione), INP (ingresso), OUT (uscita), e MEMR (lettura in memoria). Insieme alle uscite DBIN e $\overline{WR}$ del chip 8080A questi quattro segnali forniscono cinque segnali di controllo importanti che comprendono fondamentalmente la maggior parte del bus di controllo sul microcomputer MMD-1:

Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

STATUS INFORMATION DEFINITION

| Symbols | Data Bus Bit | Definition |
|---|---|---|
| INTA* | $D_0$ | Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active. |
| $\overline{WO}$ | $D_1$ | Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed. |
| STACK | $D_2$ | Indicates that the address bus holds the pushdown stack address from the Stack Pointer. |
| HLTA | $D_3$ | Acknowledge signal for HALT instruction. |
| OUT | $D_4$ | Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active. |
| $M_1$ | $D_5$ | Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction. |
| INP* | $D_6$ | Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active. |
| MEMR* | $D_7$ | Designates that the data bus will be used for memory read data. |

*These three status bits can be used to control the flow of data onto the 8080 data bus

STATUS WORD CHART

TYPE OF MACHINE CYCLE

| DATA BUS BIT | STATUS INFORMATION | INSTRUCTION FETCH ① | MEMORY READ ② | MEMORY WRITE ③ | STACK READ ④ | STACK WRITE ⑤ | INPUT READ ⑥ | OUTPUT WRITE ⑦ | INTERRUPT ACKNOWLEDGE ⑧ | HALT ACKNOWLEDGE ⑨ | INTERRUPT ACKNOWLEDGE WHILE HALT ⑩ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | INTA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $D_1$ | $\overline{WO}$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| $D_2$ | STACK | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $D_3$ | HLTA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $D_4$ | OUT | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $D_5$ | $M_1$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $D_6$ | INP | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $D_7$ | MEMR | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

Ⓝ STATUS WORD

Table 2-1. 8080 Status Bit Definitions

$\overline{\text{MEMR}}$. Lettura in memoria. Usato per fornire segnali di abilitazione ai dati da un chip di memoria verso il microprocessore 8080A.

$\overline{\text{MEMW}}$. Scrittura in memoria. Usato per fornire segnali di abilitazione alle uscite dei dati dal chip 8080A verso la memoria di lettura/scrittura.

$\overline{\text{IN}}$. Ingresso. Usato per fornire segnali di abilitazione ai dati presentati da un dispositivo esterno verso l'accumulatore all'interno dell'8080A.

$\overline{\text{OUT}}$. Uscita. Usato per fornire segnali di abilitazione ai dati dall'accumulatore verso un dispositivo di uscita esterno all'8080A.

$\overline{\text{INTA}}$. Riconoscimento d'interruzione. Usato per fornire segnali di abilitazione ad un'istruzione ad un byte presente nel registro istruzione verso il microprocessore 8080A durante un'interruzione.

Gli altri segnali collegati con il bus di controllo sono RESET, INT (richiesta d'interruzione), e INTE (abilitazione all'interruzione). Questi otto segnali di controllo vi permettono di leggere e scrivere nella e dalla memoria e nei e dai dispositivi di ingresso-uscita. Vi permettono anche di elaborare le interruzioni.

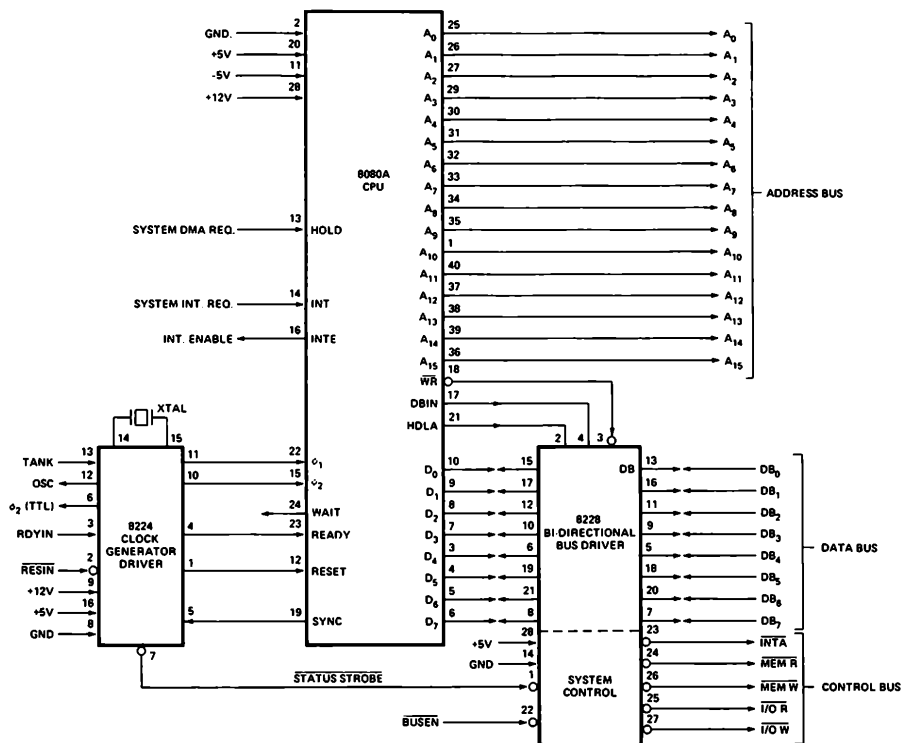
È presente un controllore di sistema e bus driver, l'Intel 8228, che realizza un buffer sul bus di dati bidirezionale nonché il latch ed il gate dei segnali di stato. Nella pagina seguente vedrete un tipico circuito a interfaccia, fornito dalla Intel Corporation. Una caratteristica dell'8228 è che esso può generare tre impulsi $\overline{\text{INTA}}$ in sequenza durante una richiesta d'interruzione; questa caratteristica è richiesta quando tentate di fornire segnali di abilitazione ad un'istruzione a due o tre byte nel registro istruzioni. Un problema del chip 8228 è che è costoso. La realizzazione del buffer sul bus di dati è limitata ad un fan-out standard di 10 carichi TTL o un assorbimento di corrente di 16 mA.

**MEMORIA**

La sezione di controllo necessaria per il chip 8080A, compresi i latch di stato e i bit di stato associati, è già stata trattata. Siamo ora pronti per aggiungere dispositivi esterni al microcomputer MMD-1. I primi dispositivi necessari sono le memorie. Esistono in varie forme e tipi, ma ne considereremo solo due, la memoria di lettura/scrittura e la memoria programmabile a sola lettura (EPROM), sono entrambe memorie ad accesso casuale. Accesso casuale significa che qualunque singola locazione di memoria può essere raggiunta dopo un'altra qualunque locazione.

Abbiamo scleto il chip di memoria di lettura/scrittura 2111 (o 8111), le cui specifiche sono date nella pagina seguente, dato che è facile l'interfacciamento con l'8080A. È organizzato in 256 locazioni di memoria, ognuna con quattro bit per locazione, cioè è una memoria a 1024 bit. L'8111-2 ha delle linee comuni di ingresso e di uscita (I/O) sulle quali vengono trasferiti i dati da e verso il microprocessore 8080A. Chiaramente, queste linee di I/O sono bidirezionali. Ogni chip di memoria 811-2 ha otto ingressi d'indirizzo (da A0 a A7) per definire in modo unico una sola locazione di memoria fra le 256 possibili. Gli ingressi di controllo all'8111-2 comprendono l'ingresso di lettura/scrittura (R/W), due ingressi di abilitazione del chip ($\overline{CE_1}$ e $\overline{CE_2}$), e un ingresso di disabilitazione dell'uscita (OD).

Dato che la lunghezza di parola in ogni chip 8111 è di soli quattro bit, coppie di tali bit devono essere abilitate e disabilitate simultaneamente allo scopo di fornire la parola a 8 bit richiesta dal microprocessore 8080A. La Figura A5-4 dimostra che $\overline{\text{MEMW}}$ (scrittura in memoria, o $\overline{\text{MW}}$) è collegata al pin 16, e che $\overline{\text{MEMR}}$ (lettura in memoria, o $\overline{\text{MR}}$) è collegata al pin 9 sulla memoria di lettura/scrittura 8111.

## SCHOTTKY BIPOLAR 8228

8080A CPU Standard Interface

# Silicon Gate MOS **8111-2**

## 1024 BIT (256 x 4) STATIC MOS RAM WITH COMMON I/O AND OUTPUT DISABLE

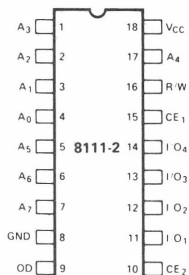
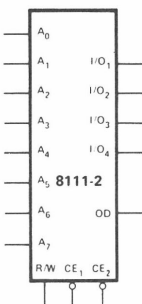
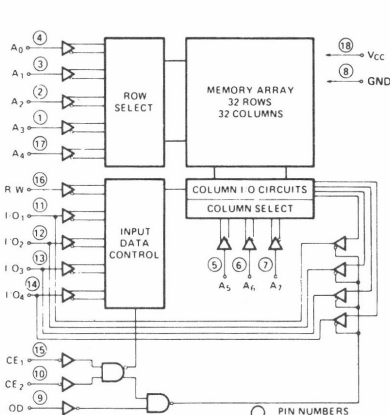
- **Organization 256 Words by 4 Bits**
- **Access Time — 850 nsec Max.**
- **Common Data Input and Output**
- **Single +5V Supply Voltage**
- **Directly TTL Compatible — All Inputs and Output**
- **Static MOS — No Clocks or Refreshing Required**
- **Simple Memory Expansion — Chip Enable Input**
- **Fully Decoded — On Chip Address Decode**
- **Inputs Protected — All Inputs Have Protection Against Static Charge**
- **Low Cost Packaging — 18 Pin Plastic Dual-In-Line Configuration**
- **Low Power — Typically 150 mW**
- **Three-State Output — OR-Tie Capability**

The Intel® 8111-2 is a 256 word by 4 bit static random access memory element using normally off N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 8111-2 is designed for memory applications in small systems where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. Separate chip enable ($\overline{CE}$) leads allow easy selection of an individual package when outputs are OR-tied.

The Intel® 8111-2 is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

Intel's silicon gate technology also provides excellent protection against contamination. This permits the use of low cost silicone packaging.

| | |
|---|---|
| $A_0$-$A_7$ | ADDRESS INPUTS |
| OD | OUTPUT DISABLE |
| R/W | READ/WRITE INPUT |
| $\overline{CE}_1$ | CHIP ENABLE 1 |
| $CE_2$ | CHIP ENABLE 2 |
| $I/O_1$-$I/O_4$ | DATA INPUT/OUTPUT |

Supponendo che il chip sia abilitato, la tabella della verità da applicare è la seguente:

| $\overline{MEMW}$* | $\overline{MEMR}$* | R/W* | OD | |
|---|---|---|---|---|
| 0 | 0 | — | — | Condizione d'ingresso non possibile |
| 0 | 1 | 0 | 1 | Scrittura in memoria; disabilitare l'uscita di memoria |
| 1 | 0 | 1 | 0 | Lettura in memoria |
| 1 | 1 | 1 | 1 | Disabilitare l'uscita della memoria |

* Nota: $\overline{MEMW}$ è identico a $\overline{MW}$ e $\overline{MEMR}$ è identico a $\overline{MR}$ nella Figura A5-4.

La decodifica del bus degli indirizzi è mostrata in Figura A5-4 sul lato destro del chip 8080A. La tabella della verità per i bit del bus d'indirizzo da A0 a A15 è:

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | . . . | A0 | Commento |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | | X | Blocco 0 (riservato alla EPROM KEX) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | | X | Blocco 1 (riservato alla EPROM) |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | | X | Blocco 2 (riservato alla memoria 8111 R/W) |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | | X | Blocco 3 (riservato alla memoria 8111 R/W) |

Qui una X indica che è permesso un livello logico o 0 o 1. Da A0 ad A7 possono essere qualunque combinazione di stati logici 0 e 1, un totale di 256 combinazioni diverse. Osservate che cambiano solo i bit d'indirizzo da A8 ad A9, dando tutte le possibili combinazioni per i due bit. I bit d'indirizzo da A10 ad A15 rimangono a livello logico 0 per tutti gli indirizzi selezionati nel nostro microcomputer MMD-1.

È d'uso decodificare in assoluto le locazioni di memoria, cioè assicurare che tutti e sedici i bit partecipino alla decodificazione di una locazione di memoria, Per il chip 8111, ciò si può fare usando i bit da A8 a A15 per fornire l'ingresso di abilitazione del chip $\overline{CE}$ desiderato. (Vedere la Figura A5-4). Dato che i bit d'indirizzo da A10 a A15 rimangono a livello logico 0, usiamo gli invertitori a collettore aperto 74LS05 in una configurazione "Wired-OR" per fornire una condizione logica decodificata in modo unico. Osservate la presenza di un resistore pull-up a 1 kΩ, R4. La tabella della verità per il circuito Wired - OR è la seguente:

| A15 | A14 | A13 | A12 | A11 | A10 | Q |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | X | X | X | X | 1 | 0 |
| X | X | X | X | 1 | X | 0 |
| X | X | X | 1 | X | X | 0 |
| X | X | 1 | X | X | X | 0 |
| x | à | x | x | x | x | ù |
| à | x | x | x | x | x | ù |

Nota X = stato logico 0 o 1

Osservate che questa tabella della verità, sebbene implementata con invertitori a collettore aperto, è identica a quella del gate NOR a 6 ingressi. Lo stato logico unico è Q = 1, e questa condizione di uscita ha luogo quando tutti gli ingressi sono a livello logico 0.

In qualunque momento l'uscita del circuito Wired-OR è a livello logico 1, sappiamo che A10 fino a A15 sono a livello logico 0 e che l'indirizzo di memoria deve essere all'interno di uno dei quattro blocchi di memoria selezionati di 256 byte. Questo si ottiene con l'aiuto di un chip di decodifica 74LS155, di cui di seguito diamo lo schema a blocchi e la configurazione dei pin,

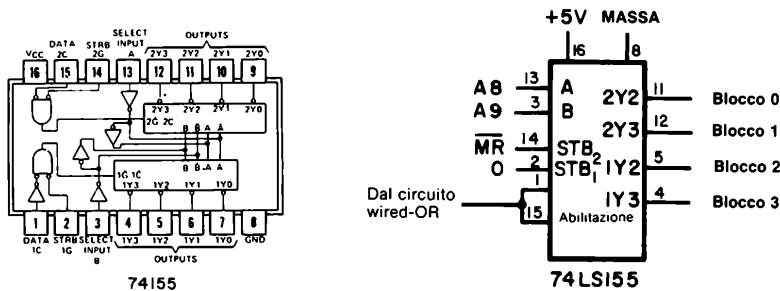



*Figura A5-5.* Configurazione dei pin del chip 74155 e schema a blocchi del circuito MMD-1.

Il chip 74LS155 è abilitato e disabilitato usando l'uscita del circuito Wired-OR, dove disabilitato corrisponde al livello logico 0 (nessun blocco selezionato) ed abilitato corrisponde al livello logico 1 (uno ed uno solo blocco di memoria selezionato). La tabella della verità per il chip 74LS155 è la seguente:

| ENABLE | MR | B | A | 2Y2 | 2Y3 | 1Y2 | 1Y3 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | X | X | 1 | 1 | 1 | 1 | Nessuna selezione |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Blocco 0 (leggere la memoria EPROM |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | Blocco 1 (leggere la memoria EPROM |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | Blocco 2 (leggere la memoria R/W) |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | Blocco 3 (leggere la memoria R/W) |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | Nessuna selezione |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | Nessuna selezione |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | Blocco 2 (scrivere nella memoria R/W |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | Blocco 3 (scrivere nella memoria R/W |

Le uscite dei blocchi 2 e 3 vanno agli ingressi $\overline{CE_1}$ (pin 15) delle rispettive coppie di chip di memoria di lettura/scrittura 8111, come si può vedere per il Blocco 3 nella Figura A5-4. Quando il pin 15 del chip 8111 è a livello logico 0, il chip è abilitato dato che $CE_2$ è cablato sul livello logico 0.

In aggiunta alla memoria di lettura/scrittura, il microcomputer MMD-1 contiene anche alcune memorie programmabili di sola lettura (EPROM). I contenuti dei chip EPROM non vengono azzerati quando togliamo l'alimentazione, come nel caso della memoria di lettura/scrittura che viene chiamata *memoria volatile.* Potete acquistare uno speciale dispositivo elettronico (o un circuito per il vostro microcomputer) chiamato programmatore EPROM e programmare i chip EPROM per le vostre speciali applicazioni.

Usiamo i chip EPROM 1702A (o 8702A) della Intel Corporation, che possono essere cancellati con l'uso di raggi ultravioletti e riprogrammati fino a quaranta o cinquanta volte. La configurazione dei pin del chip 1702A/8702A è mostrata in Figura A5-6.

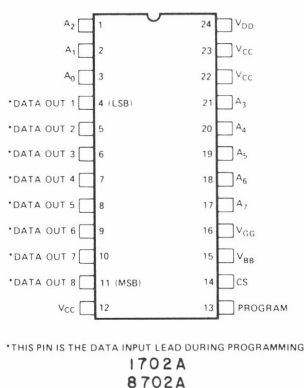



*Figura A5-6.* Configurazione dei pin del chip 1702A/8702A. Alcuni dei pin di alimentazione si usano solo durante la programmazione. Supporremo qui che il chip sia stato programmato nel modo giusto prima di essere incluso nel microcomputer MMD-1.

Osservate che vi sono otto ingressi d'indirizzo, da A0 ad A7, e otto pin di uscita dei dati, da DATA OUT 1 a DATA OUT 8, un ingresso di selezione del chip (pin 14) e molti pin di ingresso di alimentazione. Nella Figura A5-4, potete vedere come questo chip viene incorporato nel microcomputer MMD-1. I pin 12,13,15,22 e 23 sono tutti collegati al + 5 V. I pin 16 e 24 sono collegati a −9 V. L'uscita del blocco 0 dal chip di decodifica 74LS155 è collegata all'ingresso CS del 1702A (pin 14). Osservate che potete solo leggere il chip 1702A; è un chip di memoria di sola lettura, non un chip di memoria di lettura/scrittura.

## I BUS DEL MICROCOMPUTER MMD-1

I bus del microcomputer MMD-1 sono il bus degli indirizzi, dei dati e i bus di controllo. Il bus degli indirizzi consiste di sedici linee d'indirizzo con buffer. Nella Figura A5-4, è indicato il buffer dei bit d'indirizzo da A0 ad A7. Per ogni linea d'indirizzo si usa una coppia di invertitori, prima il 74L04 e poi il 7404. Il chip 74L04 ha un fan-in di 0,1, o 0,16 mA, ed è adatto all'uso con il microprocessore 8080A. I chip 8216 forniscono un buffer sufficiente per le otto linee del bus di dati, da D0 a D7. I gate NAND 7400 hanno ognuno un fan-out di 10, più che sufficiente per ogni linea di segnale del bus di controllo, come $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$, $\overline{OUT}$ e $\overline{INTA}$. RESET e INT sono ingressi al chip 8080A. L'uscita INTE potrebbe richiedere un buffer.

## INGRESSO/USCITA

La sezione di I/O del microcomputer MMD-1 è mostrata in Figura A5-7. Nel Capitolo n. 20 di questo libro, avete già familiarizzato con la decodifica di I/O e con l'uso dei latch 7575 e dei buffer three-state 8095. Di conseguenza, parleremo brevemente della sezione di I/O dell'MMD-1.

Allo scopo di trasferire otto bit di dati fra l'accumulatore all'interno del chip 8080A e un dispositivo di I/O, viene fornito un codice dispositivo a otto bit sui bus degli indirizzi da A0 ad A7. Per selezionare un unico dispositivo fra i 256 possibili è necessario un circuito di decodifica come quelli descritti nel Capitolo n. 17.

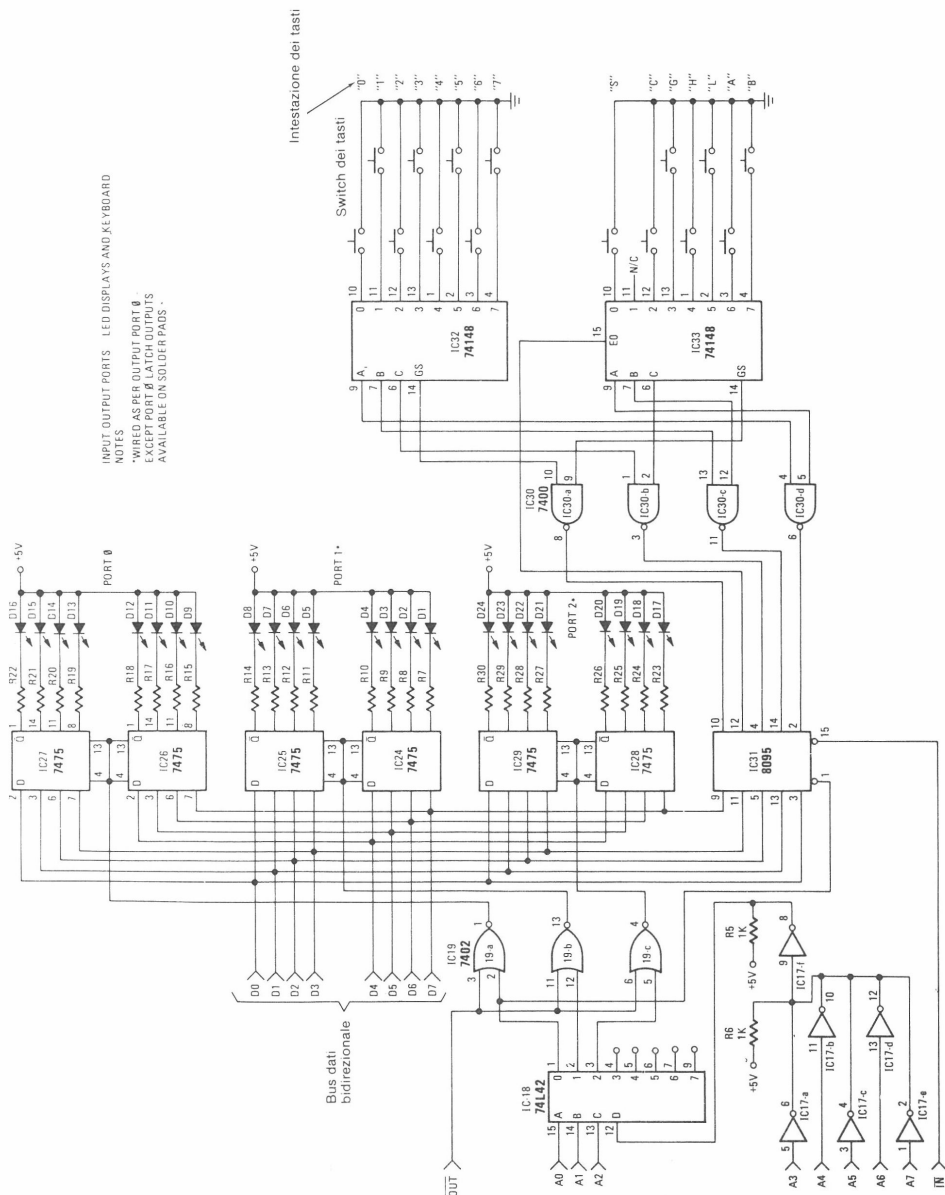



*Figura A5-7.* La sezione di ingresso/uscita del microcmputer MMD-1. Questa figura come la figura A5-4 è stampata con il permesso della rivista Radio-Electronics. Tutti i diritti riservati.

Nella Figura A5-7, il decodificatore consiste del chip 74L42 e dei sei invertitori a collettore aperto 74LS05 presenti alle linee d'indirizzo da A3 ad A7. Cinque degli invertitori funzionano come Wired-OR o gate NOR a cinque ingressi, circuito per decodificare i bit d'indirizzo da A3 ad A7 in un unico stato logico quando le cinque linee d'indirizzo sono tutte la livello logico 0. Il principio qui usato è identico a quello impiegato per la decodifica dei bit d'indirizzo da A10 a A15 nella sezione di memoria del microcomputer MMD-1. La tabella della verità è la seguente:

| A7 | A6 | A5 | A4 | A3 | Q |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| X | X | X | X | 1 | 0 |
| X | X | X | 1 | X | 0 |
| X | X | 1 | X | X | 0 |
| X | 1 | X | X | X | 0 |
| à | x | x | x | x | ù |

Nota: X = stato logico 0 o 1

Il restante invertitore a collettore aperto 74LS05 si usa per invertire Q in uno stato di livello logico 0, quando i cinque bit d'indirizzo sono tutti a livello logico 0. Questa condizione di logica 0 è applicata all'ingresso D del chip 74L42.

Il chip 74L42 è cablato come un decodificatore da 3 a 8 linee e ha la seguente tabella della verità:

| D | A2 | A1 | A0 | 0 1 2 3 4 5 6 7 | |
|---|---|---|---|---|---|
| 1 | X | X | X | 1 1 1 1 1 1 1 1 | Nessun canale selezionato |
| 0 | 0 | 0 | 0 | 0 1 1 1 1 1 1 1 | Canale 0 (tastiera e porta 0) |
| 0 | 0 | 0 | 1 | 1 0 1 1 1 1 1 1 | Canale 1 (porta 1) |
| 0 | 0 | 1 | 0 | 1 1 0 1 1 1 1 1 | Canale 2 |
| 0 | 0 | 1 | 1 | 1 1 1 0 1 1 1 1 | Canale 3 |
| 0 | 1 | 0 | 0 | 1 1 1 1 0 1 1 1 | Canale 4 |
| 0 | 1 | 0 | 1 | 1 1 1 1 1 0 1 1 | Canale 5 |
| 0 | 1 | 1 | 0 | 1 1 1 1 1 1 0 1 | Canale 6 |
| 0 | 1 | 1 | 1 | 1 1 1 1 1 1 1 0 | Canale 7 |

Viene eseguito un gate ai canali 0,1 e 2 con il segnale di controllo $\overline{OUT}$ e i canali si usano per fornire segnali di abilitazione alle informazioni dal bus di dati bidirezionale nei latch per le Porte 0,1 e 2, rispettivamente. Si esegue un gate al canale 0 con il segnale di controllo $\overline{IN}$ e lo si usa per fornire segnali di abilitazione ai dati d'ingresso presenti al buffer three-state 8095 nel microprocessore 8080A. Gli ingressi del chip 8095 consistono di due uscite da una coppia di chip di codifica di priorità da 8 a 3 linee 74148, che si usano per codificare la tastiera a 15 tasti. Compresi nella tastiera sono i tasti da 0 a 7, il tasto See/Store (S), il tasto Go (G), il tasto del byte d'indirizzo HI (H), il tasto del byte d'indirizzo LO (L) e tre tasti aggiuntivi (A,B e C) che non hanno un uso specifico. La configurazione dei pin del chip 74148 e la tabella della verità sono mostrate nella Figura A5-8. I tasti sono interruttori meccanici non ideali con logica antirimbalzo da software.

**MICROCOMPUTER MMD-1: IL SISTEMA GLOBALE**

Il sistema globale del microcomputer MMD-1 è mostrato in fotografia nella Figura A5-10, schizzato nella Figura A5-9 e come schema a blocchi nella Figura A5-11. I segnali di controllo $\overline{MR}$ e $\overline{MW}$ nella Figura A5-10 corrispondono a $\overline{MEMR}$ e $\overline{MEMW}$, rispettivamente.

SN54148, SN74148
FUNCTION TABLE

| INPUTS | | | | | | | | | OUTPUTS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EI | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | A2 | A1 | A0 | GS | EO |
| H | X | X | X | X | X | X | X | X | H | H | H | H | H |
| L | H | H | H | H | H | H | H | H | H | H | H | H | L |
| L | X | X | X | X | X | X | X | L | L | L | L | L | H |
| L | X | X | X | X | X | X | L | H | L | L | H | L | H |
| L | X | X | X | X | X | L | H | H | L | H | L | L | H |
| L | X | X | X | X | L | H | H | H | L | H | H | L | H |
| L | X | X | X | L | H | H | H | H | H | L | L | L | H |
| L | X | X | L | H | H | H | H | H | H | L | H | L | H |
| L | X | L | H | H | H | H | H | H | H | H | L | L | H |
| L | L | H | H | H | H | H | H | H | H | H | H | L | H |

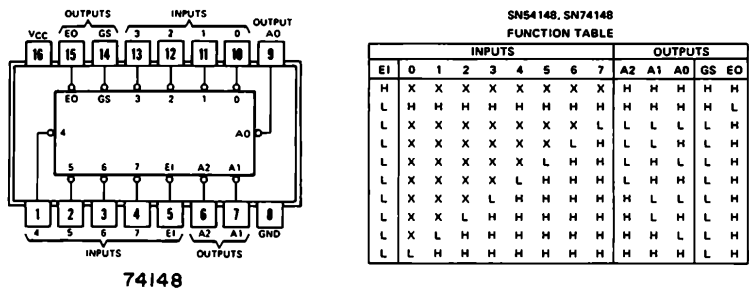



*Figura A5-8.* Configurazione dei pin e tabella della verità del chip di codifica di priorità 74148.

CLOCK
CPU 8080A
MEMORIA
REGOLATORE DI TENSIONE
DECODIFICA DELLA MEMORIA
PROM KEY
SOCKET OPZIONALE DI MEMORIA
DECODIFICA DI I/O
LOGICA DI CONTROLLO
BUFFER
PORTA 1
CONNETTORE
LED
PIN DI BREADBOARDING BP-25
ENCODER DELLA TASTIERA
TASTIERA
SWITCH ON-OFF
FUSIBILE
SOCKET DI BREADBOARDING SK-10
PORTA 0
PORTA 2

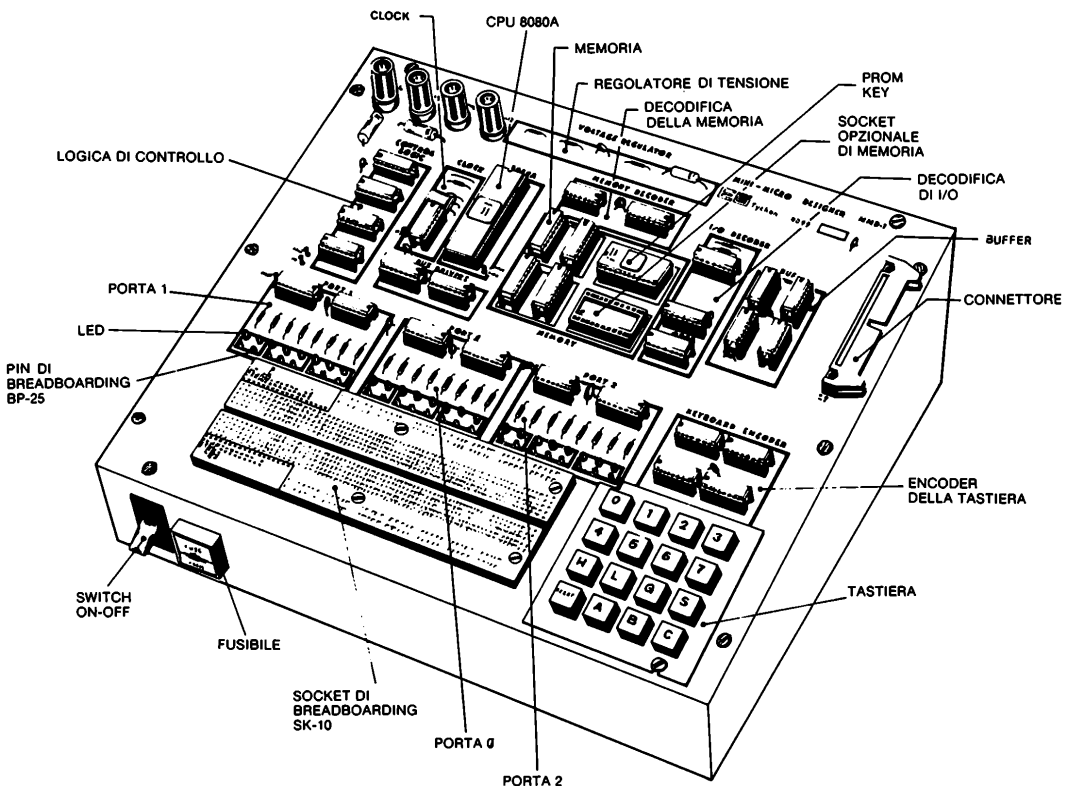



*Figura A5-9.* Disegno del microcomputer MMD-1.

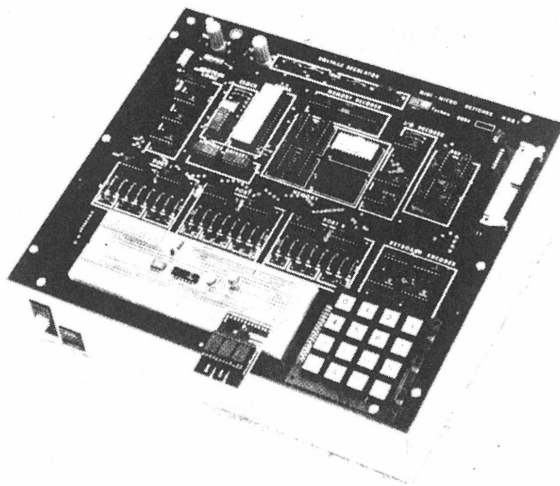



*Figura A5-10.* Fotografia del microcomputer MMD-1. Attaccato al bus della piastra vi è l'outboard bus monitor LR-27.

+5 → +5
+12 → +12
REGOLATORI DI TENSIONE
-12
-5
-9
VERSO IL SISTEMA
MEMORIA 1/2 K R/W 1/2 K PROM*
BUS INDIRIZZ HI
BUS DI INDIRIZZ. 16 BIT
BUFFER
BUS INDIRIZZ LO
READY
RESET
CONTROLLO DEL CLOCK E DELLE TEMPO-RIZZAZIONI
8080A CPU
BUS DI CONTROLLO
MEMORIA LETTURA
MEMORIA SCRITTURA
IN
OUT
$\overline{MR}$
$\overline{MW}$
$\overline{IN}$
$\overline{OUT}$
CONTROL LOGIC
BUFFER DEL BUS DATI DATA BUS BIDIREZIONALE
BUS DATI BIDIREZIONALE AD 8 BIT
DECODIFICA DI I/O
RICONOSCIMENTO DELL'INTERRUPT
LATCH
LATCH
LATCH
LEDS
LEDS
LEDS
25 BP
25 BP
25 BP
TASTIERA
* MASSIMO 1/4 K
NON TUTTI I SEGNALI SONO MOSTRATI
MINI-MICRO DESIGNER (MMD-1) BASE

*Figura A5-11.* Schema a blocchi del microcomputer MMD-1.

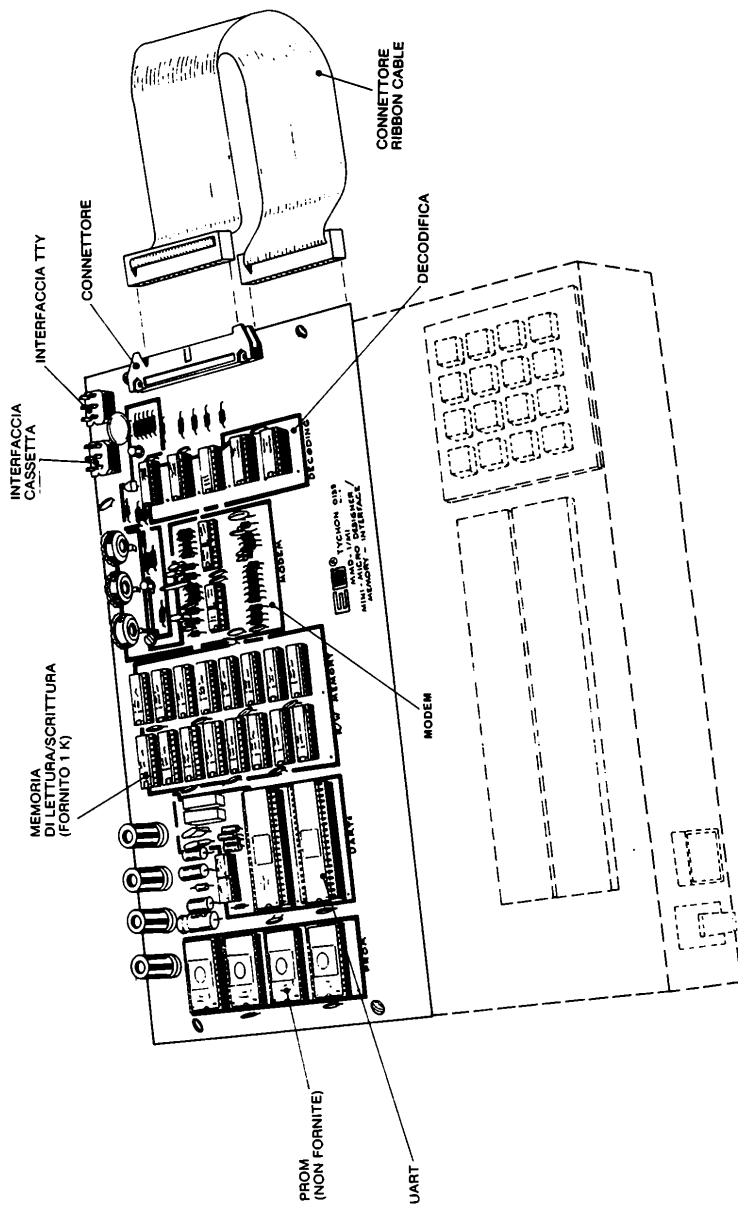



*Figura A5-12.* Piastra di interfaccia della memoria M1 che si adatta al microcomputer MMD-1.

Sebbene non ne discuteremo qui, è possibile estendere le capacità dell'MMD-1 tramite una piastra interfaccia di memoria M1, che viene posizionata sul microcomputer MMD-1 come mostra la Figura A5-12. Tale piastra vi permette di aggiungere 1 K di EPROM 1702A, 2 K di memoria di lettura/scrittura 8111-2, un'interfaccia per cassetta e un'interfaccia per telescrivente. In Figura A5-13 è rappresentato lo schema a blocchi della piastra. Per ulteriori dettagli, contattare la E. & L. Instruments, Inc, oppure la Microlem S.p.A., rappresentante in Italia della E. & L. Instruments.

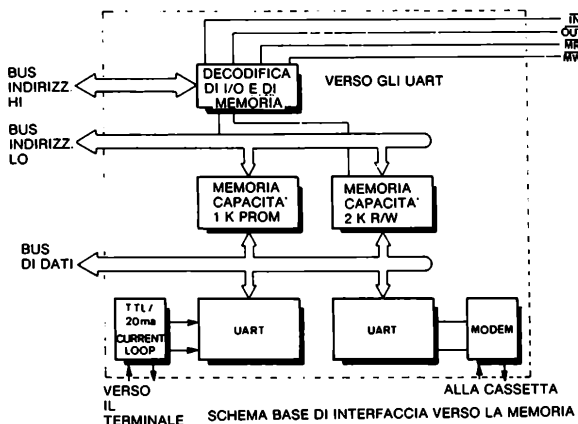



*Figura A5-13.* Schema a blocchi della piastra di interfaccia di memoria M1. La comunicazione fra il microcomputer e la cassetta o la telescrivente è in codice ASCII seriale asincrono. La conversione da parallelo in serie è attuata usando una coppia di chip UART.

## COME OPERA KEX

Il software della tastiera (Keyboard Executive software) è contenuto in una sola EPROM 1702A nella locazione stabilita per IC15 sul microcomputer MMD-1. Nelle pagine seguenti troverete un listing al riguardo. Descrivendo come opera KEX, citeremo l'articolo di J.A. Titus, dalla rivista Radio-Eletronics del Giugno 1976.

"Il software della KEX è contenuto in una sola PROM 1702A nella locazione scelta per IC15. Questa contiene tutto il software necessario per operare sulla tastiera e sui display a LED, costituenti il nostro pannello frontale controllato da software, dato che i tasti e i LED eseguono funzioni determinate dal software KEX".

"In qualunque momento venga premuto il tasto R, la CPU 8080A inizierà ad eseguire il programma dalla locazione 0. Guardando il listing del programma KEX, vedrete che immediatamente dopo aver iniziato alla locazione 0, le istruzioni fanno sì che il computer salti alla posizione HI = 000, LO = 070 (HI = 000 per tutto il programma kex) dove iniziano il programma puntando sul primo indirizzo di memoria R/W (003 000).

L'indirizzo e i dati in quella locazione sono visualizzati sulle tre porte di uscita. Ciò avviene fra POINTA e POINTC nel programma. Il software fra POINTC e POINTD eseguirà i compiti necessari per inserire nuovi dati dalla tastiera e spostare i dati sui LED. Lo spostamento viene fatto dentro l'8080A, con instruzioni opportune. Fare questo con l'hardware richiederebbe molti più circuiti integrati, ma occupa relativamente pochi passi di software."

"Le routine POINTD, POINTE, POINTF e POINTG costituiscono quello che è chiamato un decodificatore di comandi. Il software decodifica la pressione di un tasto in azioni reali. Premendo H o L fa sì che i dati vengano memorizzati temporaneamente nell'8080A ed il valore di tasti numerici è posto in uscita sul set HI o LO dei LED. Il tasto S fa sì che i dati nuovi o in corso vengano rimessi nella locazione di memoria in corso. La pressione del tasto G fa sì che il computer usi gli indirizzi HI e LO come punto d'inizio per un nuovo programma".

"Le subroutine TIMEOUT e KBRD hanno dei compiti specifici. La TIMOUT farà i suoi conteggi attraverso vari loop per circa 10 millisecondi, mentre la subroutine KBRD inserirà un codice dalla tastiera. La subroutine KBRD ha alcune caratteristiche uniche che illustrano un'interessante correlazione hardware-software. I pulsanti usati nell'MMD-1 non sono esclusi da rimbalzi, in modo che quando gli switch sono aperti o chiusi, essi possono spesso riaprire o richiudere i contatti. Questo può creare confusione al computer dato che esso non è in grado di distinguere fra una chiusura reale del tasto ed un rimbalzo. Non vogliamo che il computer avverta ogni rimbalzo come pressione dei tasti, perciò vorremmo trovare un modo di filtrarli. L'insieme dei circuiti aggiuntivi che comprende latch, clock e monostabili potrebbe fare al caso nostro, ma complica il sistema. Possiamo anche realizzare l'eliminazione dei rimbalzi per mezzo del software."

"La subroutine KBRD riconoscerà qualunque pressione dei tasti, ma non inserirà il codice dei tasti se non dopo essere sicura che il tasto è premuto effettivamente. Quanto detto è realizzato aspettando dopo aver avvertito la premuta e ricontrollando poi per essere sicura che si è in pressione. Essa controlla anche quando rilasciamo un tasto per essere sicura che il tasto ha smesso di rimbalzare prima di tentare di riscontrare che viene premuto un altro tasto dall'utente.

"I segmenti software TIMOUT e KBRD sono stati organizzati come subroutine e possono essere usati nel vostro software e negli esperimenti. Ognuna di queste subroutine può essere chiamata con un'istruzione CALL, 315. La subroutine TIMEOUT non influenza nessuno dei registri o dei flag e serve solo per ritardare il flusso del software di 10 ms."

"Una distinzione importante fra i processori 8008 e 8080 stà nell'uso delle subroutine. Nell'8008, gli indirizzi di ritorno erano memorizzati nell'8008 stesso. Nell'8080, questi indirizzi sono memorizzati in una locazione della memoria R/W. Questa è chiamata area "STACK". In qualunque momento si usa una subroutine, noi possiamo eseguirla e poi ritornare indietro al normale flusso di programma. Questi indirizzi di rientro sono molto importanti per il computer dato che forniscono l'unica istruzione di rientro fra la subroutine e il programma principale. Se vogliamo memorizzarli in una locazione di memoria R/W, il computer deve sapere dove si trova questa area di memorizzazione, se è in grado di usare correttemente gli indirizzi. Nel software KEX, quest'area viene presettata come la parte alta della memoria R/W con istruzioni alle locazioni 070, 071 e 072. L'istruzione LXI SP carica un registro "stack pointer" interno all'8080 a HI = 004 e LO = 000.

**TABLE II—KEYBOARD EXECUTIVE (KEX) PROGRAM**

```
                                *000 000
000 000  303                    JMP
000 001  070                    START
000 002  000                    0

                                /JUMP UP TO R/W MEMORY TO
                                 BE USED BY RESTARTS &
                                 VECTORED INTERRUPTS

                                *000 010
000 010  303                    JMP
000 011  010                    010
000 012  003                    003
                                *000 020
000 020  303                    JMP
000 021  020                    020
000 022  003                    003
                                *000 030
000 030  303                    JMP
000 031  030                    030
000 032  003                    003
                                *000 040
000 040  303                    JMP
000 041  040                    040
000 042  003                    003
                                *000 050
000 050  303                    JMP
000 051  050                    050
000 052  003                    003
                                *000 060
000 060  303                    JMP
000 061  060                    060
000 062  003                    003

                                /BEGINNING OF MAIN PROGRAM

                                *000 070
000 070  061      START,        LXISP    /SET STACK POINTER
                                          TO TOP OF R/W MEM.
000 071  000                    000
000 072  004                    004
000 073  041                    LXIH     /INITIAL VALUE FOR
                                          H & L
000 074  000                    000
000 075  003                    003
000 076  116      POINT A       MOVCM    /LOAD MEM DATA INTO
                                          TEMP DATA BUFFER
000 077  174                    MOVAH    /OUTPUT HI TO LED'S
000 100  323                    OUT
000 101  001                    001
000 102  175                    MOVAL    /OUTPUT LOW TO
                                          LED'S
000 103  323                    OUT
000 104  000                    000
000 105  171      POINT B,      MOVAC    /OUTPUT TEMP. DATA
                                          BUFFER DATA TO
                                          LED'S
000 106  323                    OUT
000 107  002                    002
000 110  315      POINT C,      CALL     /WAIT & INPUT NEXT
                                          KEY CLOSURE
000 111  315                    KBRD
000 112  000                    0
000 113  376                    CPI
000 114  010                    010
000 115  322                    JNC      /JUMP IF KEY WAS <
                                          010
000 116  134                    POINT D  /(0-7, OCTAL
                                          DIGIT)
000 117  000                    0
000 120  107                    MOVBA    /SAVE KEY CODE
000 121  171                    MOVAC    /GET OLD VALUE
000 122  027                    RAL      /ROTATE 3 TIMES
000 123  027                    RAL
000 124  027                    RAL
000 125  346                    ANI      /MASK OUT LEAST
                                          SIG. OCTAL DIGIT
000 126  370                    370
000 127  260                    ORAB     /OR IN NEW OCTAL
                                          DIGIT
000 130  117                    MOVCA    /PUT NEW DATA
                                          BACK INTO BUFFER
000 131  303                    JMP
000 132  105                    POINT B
000 133  000                    0
000 134  376      POINT D,      CPI
000 135  011                    011      /"L" KEY
000 136  302                    JNZ      /JUMP IF NOT AN "L"
000 137  145                    POINT E
000 140  000                    0
000 141  151                    MOVLC    /PUT BUFFER DATA
                                          IN L
000 142  303                    JMP
000 143  076                    POINT A
000 144  000                    0
000 145  376      POINT E,      CPI
000 146  010                    010      /"H" KEY
000 147  302                    JNZ      /JUMP IF NOT AN "H"
000 150  156                    POINT F
000 151  000                    0
000 152  141                    MOVHC    /PUT BUFFER DATA
                                          IN H
000 153  303                    JMP
000 154  076                    POINT A
000 155  000                    0
000 156  376      POINT F,      CPI
000 157  013                    013      /"S" KEY
000 160  302                    JNZ      /JUMP IF NOT "S"
000 161  170                    POINT G
000 162  000                    0
000 163  161                    MOVMC    /PUT TEMP. DATA
                                          INTO MEMORY
000 164  043                    INHX     /INCREMENT H & L
000 165  303                    JMP
000 166  076                    POINT A
000 167  000                    0
000 170  376      POINT G,      CPI
000 171  012                    012      /"G" KEY
000 172  302                    JNZ      /JUMP IF NOT "G"
000 173  110                    POINT C
000 174  000                    0
000 175  351                    PCHL     /GO EXECUTE PGM
                                          POINTED TO BY
                                          H & L

                                /THIS 10 MSEC DELAY
                                 DISTURBS NO REGISTERS OR
                                 FLAG

                                *000 277
000 277  365      TIMOUT,       PUSHPSW  /SAVE REGISTERS
000 300  325                    PUSHD
                                LXID     /LOAD D & E WITH
                                          VALUE TO BE
000 301  021                              DECREMENTED
000 302  046                    046
000 303  001                    001
000 304  033      MORE,         DCXD     /JUMP IN THIS
                                          LOOP UNTIL
000 305  172                    MOVAD    /D & E ARE BOTH
                                          ZERO
000 306  263                    ORAE
000 307  302                    JNZ
000 310  304                    MORE
000 311  000                    0
000 312  321                    POPD
000 313  361                    POPPSW   /RESTORE
                                          REGISTERS
000 314  311                    RET

                                /THE KBRD ROUTINE
                                 DEBOUNCES KEY CLOSURES
                                /AND TRANSLATES KEY CODES
```

```
                              /FLAGS AND REG A ARE
                               CHANGED
                              /A0-A3 = CODE; A4-A7
                               0000
000 315  333  KBRD,    IN       /INPUT FROM
                                 KEYBOARD
                                 ENCODERS
000 316  000           000
000 317  267           ORAA     /SET FLAGS
000 320  372           JM       /JUMP BACK IF LAST
                                 KEY NOT RELEASED
000 321  315           KBRD
000 322  000           0
000 323  315           CALL     WAIT 10 MSEC
000 324  277           TIMOUT
000 325  000           0
000 326  333  FLAGCK,  IN
000 327  000           000
000 330  267           ORAA
000 331  362           JP       /JUMP BACK TO WAIT
                                 FOR A NEW
000 332  326           FLAGCK   /KEY TO BE PRESSED
000 333  000           0
000 334  315           CALL     /WAIT 10 MSEC FOR
                                 BOUNCING
000 335  277           TIMOUT
000 336  000           0
000 337  333           IN
000 340  000           000
000 341  267           ORAA
000 342  362           JP       /JUMP BACK IF NEW
                                 KEY NOT STILL
000 343  326           FLAGCK   /PRESSED (FALSE
                                 ALARM)
000 344  000           0
000 345  346           ANI      /MASK OUT ALL BUT
                                 KEY CODE
000 346  017           017
000 347  345           PUSHH    /SAVE H & L
000 350  046           MVIH     /ZERO H REG
000 351  000           000
000 352  306           ADI      /ADD THE ADDRESS
                                 OF THE BEGINNING
000 353  360           360      /OF THE TABLE TO
                                 THE KEY CODE
000 354  157           MOVLA    /
000 355  176           MOVAM    /FETCH NEW VALUE
                                 FROM TABLE
000 356  341           POPH     /RESTORE H & L
000 357  311           RET

                              /THIS TRANSLATION TABLE
                               CONVERTS THE CODE
                              /GENERATED BY KEY CLOSURES
                               TO THE CODE
                              /USED BY THE MAIN KEX
                               PROGRAM
000 360  000  TABLE,   000
000 361  001           001
000 362  002           002
000 363  003           003
000 364  004           004
000 365  005           005
000 366  006           006
000 367  007           007
000 370  013           013      /S
000 371  000           000      /THIS CODE CAN'T
                                 BE GENERATED
000 372  017           017      /C
000 373  012           012      /G
000 374  010           010      /H
000 375  011           011      /L
000 376  015           015      /A
000 377  016           016      /B
```

This listing of the Keyboard Executive (KEX) is courtesy of Radio-Electronics magazine, a Gernsback publication.

Dato che il registro stack pointer è decrementato per puntare su una nuova locazione prima che venga memorizzato quanche cosa, la prima locazione dello stack sarà HI = 003, LO = 377. Controllate i vostri numeri binari a 16 bit se ciò vi sembra un pò confuso.

"Potete usare lo stack come lo organizza KEX (generalmente è una buona idea) o potete mettere il vostro stack in qualunque punto vogliate, usando l'istruzione LXISP. Ricordate di evitare l'area dello stack quando scrivete i vostri programmi. Ricordatevi inoltre che non potete mettere lo stack in un'area di memoria non esistente o in PROM.

**COME OPERA IL COMPUTER**

Vi rimandiamo al Capitolo N. 4 per una descrizione di come opera il microcomputer MMD-1. Brevemente, potete inserire i programmi dalla tastiera, esaminare i contenuti della memoria di lettura/scrittura o della EPROM, eseguire i programmi dell'8080 che sono all'interno della memoria del microcomputer e mettere in uscita le informazioni sulle tre porte di uscita. Fare tutto ciò richiede un programma memorizzato in uno dei chip EPROM 1702A, nel Blocco 0 per essere precisi. Questo chip pre-programmato, come già detto, è chiamato KEX. Quando fate partire il microcomputer 8080, prima premete il tasto RESET (R) e il microcomputer va alla locazione di memoria $0000000000000000_2$, altrimenti conosciuta come locazione 0 o HI = 000 e LO = 000. A questa locazione di memoria, il chip 8080 trova la prima istruzione che deve eseguire. Da questo punto in avanti, esiste una serie di istruzioni che funzionano come un programma di lancio iniziale per permettervi di far funzionare il microcomputer. Il programma di lancio iniziale, elencato molte pagine indietro, è solo uno dei programmi possibili. A seconda dell'uso del vostro microcomputer, potete scrivere programmi di lancio iniziale per inserire i dati da una tastiera ASCII, una telescrivente, un terminale CRT, o un nastro a cassetta. Il programma di lancio iniziale potrebbe contenere subroutine per lo scambio dei dati fra il microcomputer e lettori/perforatori o floppy disks. Descrivere tali modifiche del software va al di là dello scopo di questa Appendice. È sufficiente sapere che dovreste essere in grado di sviluppare tale software quando avete completato questo Bugbook.

# IL GRUPPO DI BLACKSBURG

I circuiti integrati a larga scala o "chips" LSI stanno creando una seconda rivoluzione industriale che ben presto ci coinvolgerà tutti. La velocità degli sviluppi in questo settore è enorme e diviene sempre più difficile stare al passo coi progressi che si stanno compiendo.

È sempre stato nostro obiettivo, come *Gruppo di Blacksburg*, creare tempestivamente e concretamente materiali didattici ed aiuti tali da permettere a studenti, ingnegneri, tecnici, ecc. di sfruttare le nuove tecnologie per le loro esigenze particolari. Stiamo facendo questo in molti modi, con libri di testo, brevi corsi, articoli mensili di "computer interfacing" e attraverso la creazione di "hardware" didattico.

I membri del nostro gruppo hanno creato la loro sede a Blacksburg, fra le montagne Appalache del sud-ovest Virginia. Mentre era in corso di preparazione attiva la nostra collaborazione di gruppo, i membri si sono occupati di elettronica digitale, minicomputer e microcomputer.

Alcune nostre precedenti esperienze:

— Il progetto e la creazione del computer Mark-8, descritto nella rivista Radio-Electronics del luglio '74, riconosciuto come primo computer per hobbisti. Si basava sul chip 8008. Da allora abbiamo progettato il Micro Designer (MD-1) e il Mini-Micro Designer (MMD-1). Di quest'ultimo computer ha riferito sempre Radio Electronics nell'articolo relativo al Dynamicro.

— La serie didattica *Bugbooks* comprende attualmente più di 15 titoli, che spaziano dall'elettronica di base, ai microcomputers, amplificatori operazionali e filtri attivi. In ogni libro vi sono esperimenti ed esempi. Siamo convinti dell'utilità di esperimenti ed esempi per rafforzare i concetti base. Questi libri sono stati tradotti anche in cinese, giapponese, tedesco e italiano.

— Siamo stati pionieri nell'uso di piccoli computers. La nostra linea di *OUTBOARDS®* rende il progetto e la prova dei circuiti molto più semplice di quanto è stato possibile nel passato. Il nostro hardware didattico viene commercializzato dalla E & L Instruments Inc. - Derby, CT 06418 Usa.

— I nostri programmi di corsi di breve durata sono stati presentati in tutto il mondo. Vengono offerte due serie di programmi, uno attraverso la Tychon Inc. e l'altro attraverso il Virginia Polytechnic Institute and State University Extension Division. Ognuno prevede esperimenti con l'elettronica digitale, e l'hardware e software dei microcomputer.
Sono previsti inoltre Continuing Education Units (CEUs). Vengono presentati corsi a gruppi aperti, società, scuole ecc.
Siamo convinti sostenitori dell'utilità degli esperimenti in questi corsi. Molto tempo perciò viene dedicato alla parte laboratorio.

— I nostri articoli mensili sull'"Interfacciamento dei Microcomputer" appaiono in quattro pubblicazioni specializzate a larga diffusione come pure in quattro periodici d'oltremare, che raggiungono circa 750.000 lettori.
Gli articoli vengono tradotti anche in tedesco e italiano.

Oltre alle attività didattiche, i nostri membri si sono occupati dello studio di software per microcomputer e automazione scientifica di strumentazione.

Mr. David Larsen e il dr. Peter Rony fanno parte della facoltà dei Dipartimenti di Chimica e Ingegneria Chimica del Virginia Polytechnic Institute e State University. Mr. Jonathan Titus e il dr. Christopher Titus fanno parte della Tychon Inc., tutti di Blacksburg, Virginia.

Il **dr. Peter R. Rony** è professore presso il Dipartimento di Ingegneria Chimica del Virginia Polytechnic Institute & State University (USA). L'elettronica digitale ed i microcomputer giocheranno un ruolo molto importante nei controlli di processo, soggetto questo di considerevole interesse per l'ingegneria chimica. Il dr. Rony è coautore di molti altri Bugbooks e di una pubblicazione mensile, denominata Columns, riguardante l'interfacciamento dei microcomputer, che appare su *American Laboratory, Computer Design, Ham Radio Magazine,* in Usa, *Electroniker* in Germania e *Elettronica Oggi* in Italia.

**Mr. David G. Larsen** è un istruttore del Dipartimento di Ingegneria Chimica del Virginia Polytechnic Institute & State University dove svolge attività didattica a vari livelli nel campo dell'elettronica analogica e digitale. È coautore di altri Bugbook e di una pubblicazione mensile, denominata Columns, riguardante l'interfacciamento dei microcomputer. Con il dr. Rony cura una serie di corsi sui microcomputer, sotto gli auspici della Extension Division della suddetta Università. Questi corsi sono particolarmente apprezzati e seguiti da professionisti di ogni parte del mondo.

**Mr. Jonathan A. Titus** è presidente della Tychon Inc. La maggior parte della sua attività alla Tychon è concentrata sulle applicazioni di microcomputer e sui piccoli sistemi a microcomputer. Ha scritto ed ha collaborato alla realizzazione di articoli sulla strumentazione e sui microcomputer, pubblicati su riviste sia a carattere professionale che hobbistico. Jon iniziò ad interessarsi ai microcomputer nel '71 quando lavorava all'automazione di strumenti per analisi chimica. La sua prima esperienza fu il microcomputer MARK 8, che era basato sull'8008.. Nel luglio '74 venne presentato dalla rivista Radio Electronics come il primo hobbista di computer. Gli attuali interessi di Jon comprendono lo sviluppo di sitemi didattici, controllo digitale e sistemi di acquisizione dati a distanza.

JACKSON
ITALIANA
EDITRICE

RONY
LARSEN
TITUS

# il BUGBOOK VI

ESPERIMENTI INTRODUTTIVI ALL'ELETTRONICA
DIGITALE, ALLA PROGRAMMAZIONE
E ALL'INTERFACCIAMENTO DEL MICROCOMPUTER 8080A

3